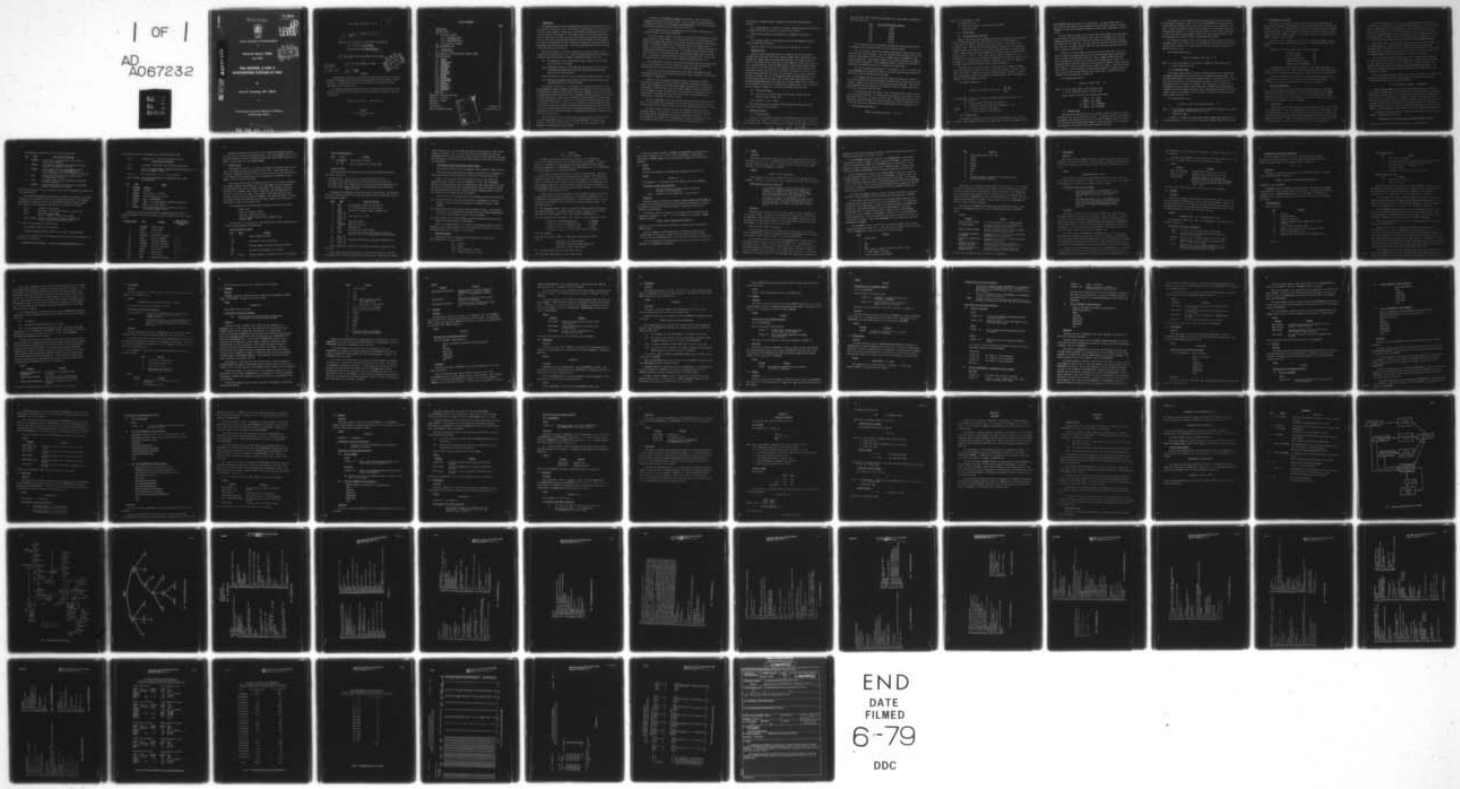
AD-A067 232 ROYAL AIRCRAFT ESTABLISHMENT FARNBOROUGH (ENGLAND) F/G 5/1
THE GEORGE 3 AND 4 ACCOUNTING SYSTEM AT RAE.(U)
JUL 78 I M CUMMINGS
UNCLASSIFIED RAE-TR-78080 DRIC-BR-66077 NL

| OF |
AD A067232



END
DATE FILMED
6-79
DDC

TR 78080

UNLIMITED

TR 78080
BR66077

LEVEL 1

ROYAL AIRCRAFT ESTABLISHMENT

*

DDC
RECEIVED
APR 10 1979
C

Technical Report 78080

July 1978

THE GEORGE 3 AND 4 ACCOUNTING SYSTEM AT RAE

by

Irene M. Cummings, BSc, MBCS

*

Procurement Executive, Ministry of Defence
Farnborough, Hants

AD A067232

DDC FILE COPY

UNLIMITED

79 04 04 036

UDC 519.688 : 681.3.061 : 657.47

(14) RAE-TR-78080

ROYAL AIRCRAFT ESTABLISHMENT

(9) Technical Report 78080

(11) Received for printing 31 July 1978

(6) THE GEORGE 3 AND 4 ACCOUNTING SYSTEM AT RAE.

by

(10) Irene M. Cummings, BSc, MBCS

(18) DRIC

(19) BR-66077

(12) 76p.

DDC
RECEIVED
APR 10 1979
C

SUMMARY

A comprehensive system for accounting for computer usage has been developed at RAE for running under the George Operating System on the ICL 1904A and 1906S computers. It allows any resource to be accounted for and has been running successfully for the last 3 years.

This Report describes the mechanism of the accounting system in detail and includes a discussion on the choice of resources actually accounted for and the charges made.

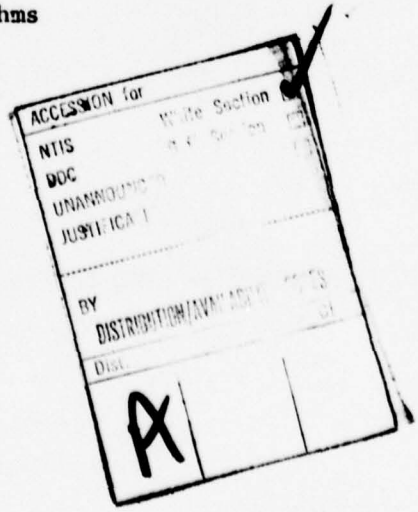
Departmental Reference: Math-Comp 233

Copyright
©
Controller HMSO London
1978

310450

LIST OF CONTENTS

		Page
1	INTRODUCTION	3
2	CHARGING POLICY	5
	2.1 Resources charged	5
	2.1.1 Filestore space	5
	2.1.2 Job resources	7
	2.1.3 Lineprinter paper	9
	2.2 Distribution of charges	10
3	BASIC ACCOUNTING DATA	10
	3.1 Journal data	10
	3.2 Filestore data	14
4	THE INTERFACE WITH THE BUDGETARY CONTROL SCHEME	16
5	ACCOUNTING MACROS	16
	5.1 JAPJOB	18
	5.2 SJPROC	19
	5.3 JAPEXPANDMAC	22
	5.4 SCANDUMP	23
	5.5 FSDICUPDATE	27
	5.6 LPCHARMAC	28
	5.7 MTLIMITS	30
	5.8 ENDFORTJOB	31
	5.9 ACCOUNTJOB	32
	5.10 MO6XMAC	33
	5.11 MO6YMAC	33
	5.12 RUNACCOUNTSA	34
	5.13 FORTPAPERMAC	37
	5.14 FILEACC	38
	5.15 NEWJAPJOBS	40
	5.16 QUARTACC	43
	5.17 QUARTUPDATE	44
	5.18 QUARTCOPY	45
6	CONCLUSION	46
Appendix A	Charging algorithms	47
Appendix B	Filestore	49
Appendix C	Budgets	50
References		52
Illustrations		Figures 1-26
Report documentation page		inside back cover



1 INTRODUCTION

Any general-purpose 'bureau' computing service needs to keep accounts of the work done for its various users. The reasons for accounting the resources used by jobs run on the computers in Mathematics and Computation Department have been argued by Sizer[1,2] and Morgan[2,3]. This Report is not so much concerned with the 'why' but the 'how' of accounting, and in particular with specific accounting jobs run under the George Operating Systems on the ICL 1904A and 1906S computers. The author's contribution has been to design the complete accounting system and design and write the software used. Both the accounting system and the software used are described in this Report.

Before proceeding to describe the methods used a brief summary of the purpose of accounting is given here. The accounting system provides information on which certain vital facets of computer management depend; these are:

(i) The budgetary control scheme; which enables users to control their rate of spend on computing.

(ii) Performance analysis; which shows the effectiveness of the choice of installation parameters and indicates those areas where desired goals are not being achieved.

(iii) Attribution of the full costs of running the 1904A and 1906S to users.

(iv) Billing of repayment users; bills are based on the quarterly accounting information sent to Finance Branch.

It has been the objective in the design of the accounting system to achieve a semi-automatic, highly reliable system which will exploit to the full the powerful facilities of the operating system, such as the automatic storage and retrieval of files and the sophisticated job control language, and which requires the minimum of human intervention. Many years of experience in the accounting of jobs under a manual operating system[4] confirmed us in the belief that human beings are much too fallible to be allowed a major role in the input to any accounting procedure. In particular, under the George Operating System the costcodes are built into the system on a semi-permanent basis rather than being submitted with every job, which would raise all the attendant problems of misreading and mistyping as was the case under the manual system.

The accounting procedure involves the running of a number of jobs, each of which is controlled by a macro (a set of George commands), at various time intervals. The time interval varies from 2 hours for job charging to three months for the quarterly accounts summary.

A mixture of programming languages has been used. Most of the early programs were written in Fortran or PLAN. Later as Algol 68 became available programs were written in this language taking advantage of the list processing and character handling facilities offered by Algol 68.

A considerable volume of data generated by two distinct sources is processed daily and is now briefly described.

The first source of data is the series of journal files generated by George, each of which is called SJFILE. As certain events occur, *eg* a job starting, a peripheral channel being released, etc, George records the details in the current SJFILE. Each SJFILE contains up to 25000 words of data on average; about 25 SJFILEs are generated every 24 hours. From this data, the 2-hourly accounting job assembles about 600 job records per day. It is obvious that with such a large volume of data to be processed each day the accounting program needs to be fail safe. An automatic restart mechanism has been built-in which will force the job to continue without loss of data after a machine breakdown during a run.

The second source of data is the series of magnetic tapes called dump tapes which contain details of every file owned by users. From this information, approximately 500 filestore charge records are generated each day.

At the end of each accounting period of two weeks, approximately 11000 accounting records are sorted, merged and listed. These listings are subsequently distributed to departments.

Because of the increasing scarcity and price of continuous computer stationery, the amount of final processed accounting information committed to paper each fortnight has been drastically pruned and other forms of output media investigated. This has led to the use of COM[5], Computer Output to Microfilm. The fortnightly accounts listings were an obvious choice for production in this form, and the accounting was one of the first computer applications in the Establishment to generate this type of output.

No major development work on the accounting system has been done since 1976 and it is anticipated that no changes, except minor modifications, will be made to the system for the duration of the life of the George Operating Systems at RAE.

The reader may be interested to know the overheads incurred by such an accounting system. An analysis of one year's running of the system shows that accounting work (including budgetary control scheme work) represents 1.5% of total computer use; computer use being measured in terms of money spent. This is an acceptable overhead in view of the value of the resulting output for

specifically accounting purposes, budgetary control work and performance analysis.

The charging policy, in respect of resources charged for and the distribution of charges between these resources, is described in section 2.

The source data used in the calculation of charges is described in section 3.

The interface between the accounting system and the budgetary control scheme is described in section 4.

Each macro used in the accounting system is described in section 5.

2 CHARGING POLICY

A job run under George must make use of some or all of the resources available such as the CPU, main store, magnetic tapes, basic peripherals, disc storage, the communications processor and stationery. As no job is ever run alone in the machine it must share and at times compete for resources with other jobs. A major objective of the charging policy is that a user should be made aware that he is not the sole user of a computer, that it costs real money to support his activity, and that he is accountable for whatever resources he uses. The other major objective is to meet the requirement laid upon the Computing Service to 'recover' the cost of running the 1904A and 1906S computers in the form of cost attributions to users; this is most easily and fairly done by devising charging algorithms that equate the notional income from the charges for use of resources to the all-in or comprehensive running costs. There are three types of charges a computer user may incur at present, and they are:

(i) filestore charges;

(ii) job charges; includes charges for CPU time, connect time and magnetic tape loadings; and

(iii) lineprinter paper charges.

The algorithms used for computing these charges are described in this section and summarised in Appendix A.

2.1 Resources charged

2.1.1 Filestore space

Filestore (see Appendix B), which is the collection of all users' files and system files, has grown at a phenomenal rate since the introduction of George on
80 the 1904A in 1971. To illustrate the rate of increase the following table shows

79 04 04 036

the size of the 1906S filestore (which began life on the 1904A) in December of each year from 1972:

Year	Size in 2048-character blocks
1972	93768
1973	216321
1974	384439
1975	601298
1976	810879
1977	1594136

It is necessary for any installation running the George Operating System to attempt to curb filestore growth for the following reasons:

(i) As filestore grows then a smaller percentage of files are able to be held on-line (on disc storage). Jobs are therefore more frequently held up while the files they require are retrieved, *ie* brought on-line from magnetic tapes known as dump tapes. One possible remedy is to increase the number of magnetic discs and disc controllers. This equipment however is costly in itself and is attended by consequential costs for increased air conditioning.

(ii) As on-line filestore grows, it is necessary to call in a special job called UNJAMMER more frequently in order to free disc space to make way for yet more files, thus increasing system overheads.

(iii) Dumping and dump processing take longer as directories increase in size, and eventually the situation would be reached when it would not be possible to make a complete copy of all the directories onto one dump tape. This would make impossible the reconstitution of filestore (known as a general restore) in the event of a serious system failure.

It follows then, that the overall effect of allowing the filestore to grow in an uncontrolled fashion, is to degrade the performance of the computers, which must affect all users. As there is no easy way of implementing filestore space rationing, users are charged for filestore space as a means of making them aware of the space they are occupying and encouraging them to erase redundant files.

The charging algorithm is:

charge per directory per day = Nn + Ss

where N = total number of files

n = charge per file

S = total size of files in terms of 512 word blocks

s = charge per block.

2.1.2 Job resources

(a) Central processor unit time

George, a multi-job operating system, provides 26 urgency levels for running jobs, the urgency levels being identified by the letters of the alphabet. A identifies the highest, most favourable level, and Z the lowest, least favourable level. The job scheduler discriminates in favour of a job running at a higher urgency level against a job running at the same time at a lower urgency level. At RAE, users are allowed budgets of CPU time in units of seconds for two urgency levels J and Q . Each job by default starts at urgency M , and continues at that level until a program is loaded then the user may elect to switch to urgency J or automatically drop down to urgency Q (if the Q budget is overdrawn then George automatically runs the job at urgency Z).

Obviously, there is a finite amount of CPU time to be shared out amongst users; two methods of control are used to limit this resource. Firstly, an upper limit of CPU time is imposed on each job running during the daytime. Secondly, users are charged for CPU time according to urgency levels used. The higher the urgency level, the higher the charge.

The charging algorithm is:

$$\text{charge for CPU time used by a job} = \sum_{u=A}^{Z} \frac{M_u C_u}{i} \quad ,$$

the symbol $\sum$ is here used to denote a summation through the letters of the alphabet;

M_u = total seconds of CPU time at urgency u ,

C_u = charge per second of CPU time at urgency u , and

either i = 1 if job started in peak hours,

or i = 2 if job started in off-peak hours.

(b) Connect time

Installation parameters are used to control the number of jobs in the
080 machine. These parameters are used to impose upper limits on the number of MOP

and BACKGROUND jobs which may be run concurrently. The upper limits were set after careful investigation and consideration with the aim of achieving optimum throughput of jobs and acceptable response at a MOP console depending on the time of day.

For the 1906S, the upper limit for MOP jobs is well below the actual number of MOP consoles (either connected via direct lines or telephone lines into the computer) which may access the 1906S, so of course not all MOP users can access the computer at the same time. For the 1904A, the situation is rather different; there are only a few MOP consoles linked into the 1904A and therefore users often have to queue for a MOP console.

It was once a fairly common occurrence for certain MOP users to log in at start of work in the morning and log out just before going home in the evening and to do very little MOP work in between. Thus, a number of MOP slots were almost permanently occupied by such users to the disadvantage of others. To encourage users to make efficient use of MOP time and to prevent a user monopolising a console all day, two controls were brought in. Firstly, a MOP job which has been 'timed out' for 10 minutes is automatically abandoned, *ie* George forces the job to log out thus freeing a MOP slot for another user. Secondly, users are charged for the elapsed time of a MOP job.

The charging algorithm is:

$$\text{charge for connect time} = Ee$$

where E is the total connect time in minutes, and
e is the charge per minute of connect time.

Connect time is defined as the elapsed time between:

LOGIN and LOGOUT
or CONNECT and LOGOUT
or LOGIN and DISCONNECT
or CONNECT and DISCONNECT .

(c) Magnetic tape

Whenever a job requests the use of a magnetic tape the request is routed by George to the central operator's console. An operator passes on the request to the magnetic tape librarian who locates the required magnetic tape (the library currently holds approximately 8000 magnetic tapes) and gives it to the operator for loading on an available and suitable magnetic tape deck.

The operator's task is made difficult because there is no way a request can be anticipated in advance, unless it is from a BACKGROUND job initiated at the central site when magnetic tape requests are specified on the accompanying job form. In such cases magnetic tapes can be brought into the machine room ready to be loaded when required. It can be appreciated that if even a small percentage of jobs in the system demand magnetic tapes, then the operators are kept very busy meeting these requests as well as performing other duties.

The handling of magnetic tapes under George therefore presents operational overheads and for this reason users are encouraged to use this resource with discretion by the imposition of a fixed charge for every successful onlining of a magnetic tape to a job.

The charging algorithm is:

$$\text{charge for magnetic tapes used} = Tt$$

where T is the total number of occurrences of magnetic tapes onlined, and
t is the charge per online.

2.1.3 Lineprinter paper

On average, approximately 50 boxes of lineprinter paper are consumed each fortnight by lineprinters in the central site (including the Maths 7020). One box holds 2000 sheets of continuous lineprinter paper. Because of the increasing price and the difficulty in obtaining stocks of paper, it is the policy of Computing Division to control the demand. The correctness of this policy is confirmed by a recent Defence Council Instruction (T80/76) which requests that "computer users should ensure that their use of continuous stationery for printed output is kept to a minimum". Hence, a charge is levied for each page of lineprinter paper output by a job.

The charging algorithm is:

$$\text{charge, per user, for lineprinter paper} = L\ell$$

where L = total number of pages output on the central site lineprinters (including the Maths 7020) by the user's jobs,

ℓ = charge per page.

In time, it is hoped that applications which consume large amounts of continuous stationery will be converted to using COM (computer output to microfilm).

2.2 Distribution of charges

The values of the charging coefficients for the various algorithms are derived after studying the performance of the computer under normal workloads over a period of time. The study determines: the size of filestore in terms of number of files and blocks, average connect time per hour, average mill time at various urgencies per hour, number of pages of lineprinter paper per accounting period, number of magnetic tape loadings.

The installation manager has to decide how to weight the charges, that is, the proportion of the required return per hour to be contributed by each item above.

Computing Division has chosen the following weightings:

filestore charges	59%
CPU time charges	20%
connect time charges	10%
lineprinter paper charges	10%
magnetic tape load charges	1%

The weighting for each resource reflects, to some extent, the degree of control required for its use. The relation between demand and available supply, and the costs of increasing the supply are relevant factors in the consideration.

Having established the return per hour required and the distribution of charges then it is possible to set values to the charging coefficients. A list of charges in force since 28 February 1977 is given in Appendix A.

3 BASIC ACCOUNTING DATA

In this section, the data used by the accounting system is described. There are two sources of data used. Firstly, the data stored in the JOURNAL files; this is used in the calculation of job and lineprinter paper charges. Secondly, the filestore directory files, as recorded on a 'dump' tape; this is used in the calculation of filestore charges.

3.1 Journal data

As George processes jobs, messages of various types are sent out to inform the 'outside world' of certain events which have occurred. In this Report, only a brief description of the central message system will be given and readers are referred to the George 3 and 4 Operational Management Manual[6] for a more comprehensive description. However, the information given here will be sufficient for an understanding of how George provides data for an accounting program.

An 'event' message may fall into one or more categories in the sense that it may be sent to one or more of several possible destinations, such as the central console, a MOP console or a monitoring file. The possible destinations of each message are determined by certain bits being set in a 'category' word which is part of each message; however, the message may not actually appear at a specified destination because a user is able to control which types of messages are sent to his monitoring file or MOP console by using the TRACE or REPORT commands.

The category that is of particular importance to the accounting program is the JOURNAL category. Messages in this category are routed to the current :JOURNAL.SJFILE. Each SJFILE, which is a basic file, holds 2000 to 3000 messages and approximately 25 SJFILEs are filled every 24 hours. For brevity, the series of :JOURNAL.SJFILEs will be often referred to simply as the Journal.

There are several hundred message types in the internal central message directory which may be sent out. Each message has a unique message identifier, *eg* JOBT identifies the 'job finished' message, and a message skeleton. The message skeleton is made up of invariant text and parameter identifiers. The form of the parameter identifiers depends on whether the message skeleton is 'unpacked' or 'packed'.

Unpacked message parameter identifiers consist of %A, %B, etc, and when the message is assembled for output, simple parameter substitution takes place, *eg* the message skeleton for the message JOKAL is

OK: YOUR ALLOWANCE IS NOW %A : CONSUMED%B .

Packed message parameter identifiers indicate the position of specific data within a message and each is followed by a character, known as the parameter description character (PDC), from the ICL 1900 64 character set, indicating the type of data to be substituted. Each PDC has a name, *eg* PDC 8 has the name JOBTYPE, signifying that the data associated with PDC 8 indicates the type of job. Some PDCs simply describe the type of quantity they refer to, *eg* PDC A has the name NUMA, signifying that the data associated with this PDC is a one-word binary number. A packed message is indicated by bit 3 of the category word being set.

An example of a packed message is JTYPE, sent out when the type of a job changes. The message skeleton is:

%A2%B+JOBTYPE IS NOW %C8%D(,CONSOLE PROPERTY:%D#%E(,CLOCKED%E= .

There are six PDCs in this message and they are 2,+,8,(,# and = .

The meaning of each PDC is shown in the following table:

PDC	Name	Data format and meaning
2	TIMENOW	1 word binary; time expressed as the number of seconds since midnight
+	JOBMILL	1 word binary; cumulative mill time for this job expressed in hundredths of seconds
8	JOBTYPE	1 word binary; there are four possible values: 0 for a MOP job; 1 for a BACKGROUND job; 2 for a remote MOP job; 3 for a remote job entry job
(	SKIP	0 length; indicating that the following part of the message may be omitted
#	PROPS	a variable length string of characters indicating the console property
=	PROGMILL	1 word binary; mill time clocked so far by program measured in hundredths of seconds

Before this message is sent out by George, the relevant information is substituted for the parameters.

Both packed and unpacked messages may be routed to the Journal. For packed messages, only the parameter parts of the messages are sent out. Any program analysing the Journal must decode the message by consulting a table of PDCs. The format of a packed message as written to the Journal is as follows:

word 0	category of the message
word 1	job number if the message related to a user's job
word 2	bits 0-11 message number bits 12-23 number of PDCs in the message
word 3 onwards	string of PDCs directly followed by appropriate data for each PDC in same order as PDCs

The record is terminated by a checksum word.

As an example, suppose that the message

```
STARTED :PERFECAL,T-M47SM, 4OCT76 11.51.02 TYPE : BACK UO.LO(*TR)
```

is sent to a monitoring file. The message would be in the following packed form when sent to the Journal:

```
<2D@05MX1E0837128-AEPERFECAL  T-M47SM  06V40:*F0001000000000002*TR (LXV  .
```

The different parts of the message may be interpreted as follows:

word 0	category word = <2D@ which has the bit pattern 00110000001010010010000
word 1	job number = 05MX which has the decimal value 23416
word 2	bits 0-11 message number = 1E which has the decimal value 101 bits 12-23 number of PDCs = 8
words 3, 4	the actual PDCs = 3, 7, 1, 2, 8, -, A and E
word 4 onwards	the corresponding data for each PDC as shown in the following table

PDC	Name	Value
3	USERNAME	PERFECAL
7	JOBNAME	T-M47SM
1	DATENOW	06V4 (days), *ie* 4 OCT 76
2	TIMENOW	0:*F (seconds), *ie* 11.51.02
8	JOBTYPE	0001, *ie* BACKGROUND
-	GEOPER	0000, *ie* UO (geographical peripheral identifier)
A	NUMA	0000, *ie* LO (line number)
E	VARCHAR	*TR (the preceding 0002 is the count of words in the variable length string)

Following is a list of the messages (by number) which are used by the accounting program to build-up job charge records and lineprinter paper charge records.

Message number	Name	Meaning	PDCs used by the RAF accounting program
3	BMRESTART	George restart	1 2
7	GREADY	George ready	
14	JDCHAN	Date change	1
15	JCONTINUE	Saved job continuing	
73	JTYPE	Job type changed	2 8
80	JLFCOMP	Listfile completed	B 3 7
81	JLFCANTDO	Listfile interrupted	B 3 7
82	JLFTERM	Listfile terminated	B 3 7
101	ASTART	Job started	1 2 3 7 8
102	JUSURG	Urgency used	2 + :
103	JOBT	Job finished	2 +
108	JPER	Peripheral on-lined	2 ?

080

Readers are referred to the George 3 and 4 Operation Management Manual[6] for the full list of PDCs and message skeletons. Note that messages 80, 81 and 82 have been modified by RAE[7] by including an extra PDC, namely NUMB, which gives the number of pages output by the LISTFILE command.

3.2 Filestore data

As described in section 2.1.1 users are charged for filestore space according to the number and size of terminal files they own. These quantities are continually changing throughout the day as jobs are run, so an arbitrary time has to be chosen and the charging based on the composition of filestore at that moment. This is achieved as follows.

Once a day, the filestore is dumped to protect it in the event of hardware or software failures corrupting files. The files dumped are the directories, certain system files and all terminal files marked 'to-be-dumped' by George. The dump is to a magnetic tape, called a dump tape, and each dumped file is stored in a subfile on the tape. Those subfiles containing directory files provide the source data for filestore charging. Each dump of filestore is known as a dump increment and identified by an increment number; each dump tape is identified by a tape serial number (tsn).

On the magnetic tape (the dump tape) the layout of data is as follows:

Header label

Start of increment sentinel

Increment (composite subfile)

End of increment subfile (end of composite file trailer label)

Each sentinel is 20 words long. Those words which contain information used by the accounting program are shown below.

Start of subfile sentinel

Word	Value	Meaning
0	6	start-of-subfile sentinel
2, 3, 4		local name of file or directory
6		increment number in start-of-increment sentinel
13, 14, 15		username if file dumped is a directory
19	bit 0=1	indicates simple or composite subfile of a directory

08

End of subfile sentinel

Word	Value	Meaning
0	#40000000	end of subfile sentinel
	or	
	#7	end of composite file trailer label

Directory files

Each directory file contains an entry for each file owned by that user. The entry consists of

the name record (contains general information about the file),
the blocks record (gives the disc location of each block of the file),
the index record (if an indexed file; contains search key information), and
the traps record (lists those users who have access to the file and the type of access)

Information for charging is extracted from the name record. The following shows the particular words of the name record which contain information used by the accounting program.

Word	Name	Value and meaning
0	EREC	Record header, always has the value 41
1	ERES	0, to indicate this is a name record
2	ESER	The tsn of a magnetic tape, otherwise zero
3	ELOC1	Local name of the file
4	ELOC2	
5	ELOC3	
7	EGEN	Generation number
9	EWRITDAY	Date last written to
13	ELAN	Language code
29	EDLA	Date when file last accessed
37	ECOPS	Value of bits 0-8 gives the number of blocks in the file. Bit 23 = 1 indicates that file is on-line
38	EUSE1	Username if entrant is a directory, otherwise zero
39	EUSE2	
40	EUSE3	

This information is sufficient to calculate a filestore charge for each directory.

There remains just one more problem. Not all directories are associated with proper users; many are associated with pseudo users. Therefore, the lowest

80

superior proper user for each pseudo user must be found before the users' money budgets are updated. This is achieved by using the information in the file DICTIONARY which records the superior user of each user; thus the whole directory hierarchy may be reconstructed and the lowest superior proper user of each pseudo user determined.

4 THE INTERFACE WITH THE BUDGETARY CONTROL SCHEME

The accounts interface with the budgetary control scheme is shown in Fig 1.

Each accounting program generates charge records which are accumulated in files; these files are then processed at the end of the accounting period.

Every 2 hours the job accounting program calculates job charges and updates the users' budget records (see Appendix C) in the file DICTIONARY. Details of lineprinter paper used are accumulated at this stage ready to be used as data for the lineprinter paper charging program. Each day filestore charges are calculated and each week the lineprinter paper charging program is run; in both cases the users' budget records are updated.

At the end of the accounting period the following processes are initiated:

(i) The money spent by each user, as recorded in DICTIONARY, is read and stored in a file for later processing by the budgetary control scheme programs.

(ii) Each authorised user is given a fresh allowance of money. The allowance for each user is calculated by the budgetary control scheme programs based on the user's past spending and the size of his budget.

(iii) The records of the various charges are sorted, merged and listed by user-name prior to distribution to departments.

There is a two-week delay in the feedback of computed allowances from the budgetary control scheme because of the time required to process the accounting information from all the computers in the scheme.

5 ACCOUNTING MACROS

Each macro in the accounting system is described in this section. Each description of a macro includes:

(i) function
(ii) format
(iii) macro parameters (if any)
(iv) program parameters (if any)

(v) execution

(vi) error messages.

Listings of the macros described in sections 5.1 to 5.18 appear in Figs 4 to 21. These listings are of the macros used on the 1906S. The macros used on the 1904A have the same name but there are slight differences within the macros themselves, *eg* JAPEXPANDMAC on the 1906S appends to a file JAPJOBS1906S but on the 1904A to a file called JAPJOBS1904A.

Fig 2 shows the logical position of each macro in the system. Also shown are the files used for passing data from one macro to another. The interaction between the macros will become apparent in the description of the macros in this section. In order to avoid a mass of detail, descriptions of intermediate files, mainly magnetic tape files, have been omitted as they are only of interest to the programmer setting up or maintaining the system. However, it is worth saying a little about the sorting keys used.

A major part of the processing of the accounting data involves the sorting of data records into particular groupings. The sorting is performed by the standard ICL sorting programs XSDA and XSDC[8]. Certain items of data in a record are chosen as sorting keys to be used by the particular sorting program. The principle keys are:

(i) The department number. This is an arbitrary number chosen to designate a particular department. A department number of zero is given to those records which relate to non-department users such as DUMPER, OPERATORS, etc.

(ii) The hierarchical level number. This number is in the range 0 to 3 where:

0 refers to a non-department user, *eg* :SYSTEM
1 refers to a user at department level, *eg* :MATHS
2 refers to a user at a division level, *eg* :MM1
3 refers to a user at costcode level or below, *eg* :MM1164 .

(iii) The username.

(iv) The record type. This is a number which identifies the type of record.

1 indicates a job charge record,
2 indicates a filestore charge record, and
3 indicates a paper charge record.

(v) The date (start date for a job charge record).

(vi) The time (start time for a job charge record).

Three of the macros, SJPROC, LPCHARMAC and FSDICUPDATE run programs which update users' budgets held in the file called DICTIONARY. George budgets (principally the MONEY budget) and associated George commands are described in Appendix C.

5.1 JAPJOB

Function

Controls the running of the SJPROC macro (described in section 5.2).

Format

JAPJOB a,b,c .

The parameter a is mandatory and must appear as the first parameter.

Description of the macro parameters

a	the time interval, in minutes between successive runnings of the SJPROC macro
b and c	parameters for the SJPROC macro, see section 5.2

Execution

Whenever George is reloaded, the message 'PLEASE ACTIVATE JAPJOB THANK YOU' is output to the central operator's console reminding the operator to initiate the running of the Journal Accounting Program. The operator does this by invoking the JAPJOB macro.

The macro is first set waiting for 'a' minutes, then it issues the JAP command with parameters b and c . The JAP command in turn issues a RUNJOB command as follows:

RUNJOB JAJOBA,:JOURNAL,SJPROC,PARAM(b,c) .

The macro is then set waiting for a further 'a' minutes before reissuing the RUNJOB command.

The JAP command is an RAE modification to George such that the job, :JOURNAL.JAJOBA issued by it, is given the classification 'system issued', *ie* it will run as a BACKGROUND job in one of the slots reserved for system jobs.

The JAP command and the JAPJOB macro were written by Mr M.J.D. Thurston of Mathematics and Computation Department.

5.2 SJPROC

Function

This macro controls the running of the Journal Accounting Program (JAP). The input is the series of :JOURNAL.SJFILEs and the output consists of records of jobs and lineprinter paper usage. During the course of each run, user budget records in the file DICTIONARY are updated.

Format

SJPROC SJFILE a,RAEJAPREC b .

Both parameters are optional except for the very first run when both must be given. If either or both parameters are omitted then default actions are taken.

Description of the macro parameters

a — the generation number of the first SJFILE which is to be processed; the processing will then continue with SJFILE(a + 1), SJFILE(a + 2) and so on. By default, the accounting program will start with the last SJFILE being processed the last time the macro was run

b — the generation number of the RAEJAPREC file which is to be created to receive job and lineprinter paper records output by the JAP. By default, the JAP will output to the last RAEJAPREC file opened the last time the macro was run

Execution

The macro first checks that it is being run under the user :JOURNAL, with the job name JAJOBA. If either of these checks fail then the run of the macro is terminated. These precautions are to ensure that SJPROC is never run alongside another job also running SJPROC; if this should happen, users would be in great danger of being charged twice for their work!

The macro then checks the parameters.

If both parameters are given, SJPROC loads a virgin copy of the program, namely PROGRAM A68R, assigns the appropriate SJFILE and RAEJAPREC files to the JAP, and activates the JAP which proceeds to process the SJFILE data.

If the SJFILE parameter is omitted, then PROGRAM A68R is loaded which when activated reads an 'update' record in the DICTIONARY file. The program gets from this record the name of the file containing the last version of the Journal Accounting Program, *eg* RAEJAPBIN(3). The program sends a message to SJPROC to load the program which when loaded and activated informs the macro, via a HALTED message, of the SJFILE to be used. The RAEJAPREC file requested is assigned, the

program will then start processing the SJFILE data after skipping the records in SJFILE processed last time the macro was run.

If the RAEJAPREC parameter is omitted, then PROGRAM A68R is loaded which when activated reads an 'update' record in the file DICTIONARY. Stored in this record is the generation number of the RAEJAPREC file to be used next. This generation number is passed to the macro via a HALTED message. The macro then assigns the SJFILE and RAEJAPREC files and begins processing the SJFILE data.

If both parameters are omitted then PROGRAM A68R is loaded and activated; it then reads the 'update' record in the file DICTIONARY, informs the macro, via a HALTED message, of the file containing the program to be loaded. The program is then loaded and informs the macro via HALTED messages which SJFILE and RAEJAPREC files are to be used. These files are then assigned and the program then proceeds to process the SJFILE data after skipping the records in SJFILE processed last time the macro was run.

Note that the 'update' record is read by a special extracode peri type 60 mode #76, and the RAEJAPREC files are created under the pseudo user :JAPFILES.

As the SJFILE files are processed, the accounting program sends job and lineprinter paper usage records to the current RAEJAPREC file. The program also accumulates in an array details of TIME (at different urgencies) and MONEY used by each job. When TIME and MONEY details for approximately 25 jobs have accumulated then it is time to update users' budgets in the file DICTIONARY. Before this is done, however, the program is stored in a file called RAEJAPBIN. Three generations (1, 2 and 3) of this file are used cyclically and the next one in the series is chosen. The current RAEJAPREC file is released and a new file RAEJAPREC(+1) assigned.

The DICTIONARY is then updated with the accumulated budget data using the budget extracode type 60 mode #77. At the same time an 'update' record is written to the DICTIONARY file. This record has the following format:

Word	Contents
0	record header
1	2
2	*UPD
3	ATE
4	A68R - name of program creating this update record
5	1 - mark number of program
6	time in seconds since midnight

Word	Contents
7	date as days since 31 DEC 1899
8	unused
9	unused
10	unused
11	unused
12	RAEJ
13	APBI
14	N(
15	generation number of RAEJAPBIN containing the latest Journal Accounting Program
16	)

Eventually, the accounting program catches up with George, *ie* it is reading the SJFILE currently being written to by George. When the program detects that the time on the record it is processing is about 10 minutes or less behind real time, processing of the current SJFILE is terminated, the accounting program is stored in the next RAEJAPBIN file, the current RAEJAPREC file released, the users' budgets finally updated, an 'update' record written to the file DICTIONARY and then the program and macro terminate.

Note that each RAEJAPREC file produced has the ERASE trap closed. This trap on each RAEJAPREC file will be opened subsequently by the JAPEXPANDMAC macro.

Errors

Message	Meaning
WRONG USER OR JOB NAME	Job running SJPROC is not :JOURNAL.JAJOBA
NO UPDATE RECORD	No update record exists in DICTIONARY with the given program identification
ERROR IN UPDATE RECORD?	Expected information about current RAEJAPBIN file is not in the update record
THE RAEJAPREC REQUIRED NOT KNOWN	Program failed to output a message specifying RAEJAPREC file to be assigned
THE SJFILE REQUIRED NOT KNOWN	Program failed to output a message specifying SJFILE to be assigned
REQUEST FOR GEN NO. OF RAEJAPREC NOT MADE	Program failed to output a request for generation number of RAEJAPREC to be used
REQUEST FOR GEN NO. OF SJFILE NOT MADE	Program failed to output a request for generation number of SJFILE to be used

80 All the above errors will cause the macro to be terminated.

5.3 JAPEXPANDMAC

Function

This macro runs a program which takes as input the records in the series of RAEJAPREC files produced by SJPROC. These records are expanded and separated out into job and lineprinter paper usage records. The macro is generally run once a day.

Format

JAPEXPANDMAC START a, END b

Both parameters are optional except where the macro is run for the very first time; on that occasion the START a parameter must be given. If either or both parameters are omitted then default actions are taken.

Description of the macro parameters

a — the generation number of the RAEJAPREC file at which processing is to begin. By default, the next RAEJAPREC file due to be processed is chosen

b — the generation number of the last RAEJAPREC file to be processed. By default, processing continues until the next-but-one latest generation (not necessarily the highest) of RAEJAPREC has been processed

Execution

Initially, the macro, which runs under :JOURNAL, must determine which RAEJAPREC should be the first to be processed. Either the START a parameter is given or else the information is acquired by loading and activating the program which will then output a HALTED message indicating the next RAEJAPREC due to be processed. A check is then made that this RAEJAPREC is suitable for processing, *ie* it must not be the latest RAEJAPREC file (because it is likely that this will be overwritten by the next SJPROC run).

Two output files, JAPJOBS1906S (or JAPJOBS1904A) for job records and TEMPJAPLIST for lineprinter paper usage records are assigned. The RAEJAPREC file is assigned and processing commences; the records in RAEJAPREC are expanded and appended to the appropriate output files. When a RAEJAPREC file has been processed, the next generation of the file is assigned and processing continues.

Processing is terminated when either, the last RAEJAPREC file as indicated by the END b parameter has been processed, or when the next-but-one latest RAEJAPREC file has been processed. At this point, the generation number of the

next RAEJAPREC file is preserved in the program. Listings of the output files are produced.

Note that the RAEJAPREC files have their TRAP settings changed after being processed; The ERASE trap is opened and the READ trap closed.

Errors

Message	Meaning
FGN %C OUT OF RANGE	RAEJAPREC generation %C does not exist
ERROR IN FGN %C	RAEJAPREC generation %C does not exist
PROGRAM ERROR	either, program failed to request the generation number of the RAEJAPREC file to be used at the start of the next JAPEXPANDMAC run, or the program failed to halt with the 'SAVE' request

All three of the above errors will cause the macro to be terminated.

5.4 SCANDUMP

Function

This macro, invoked by the operators at the central console once a day, runs a program which performs up to three tasks simultaneously. Firstly, the calculation and output of the filestore charges; secondly, the listing of files not accessed within a certain time period; thirdly, the generation of a macro for retrieving files in the event of a general restore.

Format

SCANDUMP TAPE a, INC b, ACCOUNT, OLD c, JUG d

The parameters TAPE a and INC b are mandatory. The parameters may be given in any order.

Description of the macro parameters

TAPE a	a is the tsn of the dump tape to be processed
INC b	b is the number of the increment on the dump tape which is to be processed
ACCOUNT	indicates that filestore charges are to be computed
OLD c	indicates that a list of files not accessed in the last c days is to be output
JUG d	indicates that a macro called JUGGERNAUT is to be generated containing RETRIEVE commands for certain files accessed within the last d days

Description of the program parameters

The file ACCDATA consists of one line containing two real numbers separated by at least one space; the first number is the charge per block (in pence), the second number is the charge per file (in pence), *eg*

0.25 3.0 .

Execution

This macro is invoked by the operator at the central console, it in turn issues a RUNJOB command as follows:

RUNJOB C-M06-SCDUMP,:DUMPER,SCANDUMPJDF,PARAM

(parameters as for SCANDUMP).

SCANDUMPJDF loads an imbedded program, which when activated first checks on the validity of the increment number and *tsn* given. It does this by scanning through the file :SYSTEM.INCINDEX for the increment number. Each entry in INCINDEX has the following format (only those words of interest to the program are shown):

:SYSTEM.INCINDEX

Format of entries:

Word	Meaning
0,1	red tape
2	increment number
3	increment state word
12	number of magnetic tapes holding copies of the increment
13	tsn of the first magnetic tape
14	magnetic tape state word for first magnetic tape
15	tsn of second magnetic tape
16	magnetic tape state word for second magnetic tape
:	

and so on

Increment state word

Bit	Meaning
0	increment to be redumped
1	end of batch increment, *ie* one in which the end of filestore scan was reached without encountering end-of-tape
2	increment marked to be redumped by TTTP
4	increment obsolete
5	increment not suitable for retrieving files.

Magnetic tape state word

Bit	Meaning
0	tape failed or format error
4	tape not available

When the entry for the increment is found in INCINDEX, the tsn's for that increment are checked for the tsn given as a parameter in the TAPE a parameter. An error message is displayed and the macro abandoned if the increment is unsuitable, or if the dump tape is corrupt, unavailable or unsuitable.

If the magnetic tape and the increment number specified are valid then the magnetic tape is onlined, the program scans through the magnetic tape for the start-of-increment sentinel for the particular increment. Once the increment is found then the dumped directory files in that increment are processed.

If the ACCOUNT parameter was given, then the program totals up for each directory the number of terminal files and their size. When the end of each directory subfile is reached a charge for that directory is calculated and the username and charge are recorded in an array for later processing.

Note that certain terminal files are not charged for, they are monitoring files and the JOBLIST file.

Towards the end of the increment is the subfile containing the file DICTIONARY; from the entries in DICTIONARY, the program constructs a table in main store where each entry consists of the name of each pseudo user against the name of its superior. The name of each superior user in the table is then compared against each pseudo username in the table and if it is found to be a pseudo user itself then it is replaced by the superior username. The table is repeatedly scanned in this way until the entries in the table each consist of the name of a pseudo user against the name of its lowest superior proper user.

The usernames in the filestore charging records can now be checked against the constructed pseudo/proper user table and any pseudo user replaced by proper user names. The charge records are then output to a new generation of the file FILELIST.

080

If the OLD c parameter is given, then as each directory file is processed any terminal files which are found not to have been accessed in the last c days are recorded. The following information is produced for each such file; file description, date when file written, date when file last accessed, and the size of the file. For each directory, the total number of files not accessed in the last d days and their total size are given. These 'old-file' listings are then distributed to users in the hope that they will be encouraged to erase unwanted files. When the end of the increment is reached the grand total of 'old' files and their size is produced.

If the JUG parameter is given, then a macro called JUGGERNAUT is generated containing a series of RETRIEVE commands for files which fulfil all the following conditions:

(1) is online;

(2) has been accessed within the last d days;

(3) has the highest generation number for that file with a particular language code, *ie* if FRED(8/MKP), FRED(9/MKP), FRED(9), FRED(9/MKQ), and FRED(10) were all online and accessed within the last p days, then FRED(9/MKP), FRED(9/MKQ) and FRED(10) would be chosen.

It must be said that this facility was used only on two occasions when a general restore was necessary and proved to be a useful first step in the design of a more efficient strategy for the retrieval of files in such circumstances. This particular retrieve facility is very inefficient because the RETRIEVE commands are not sorted according to location of files on dump tapes, hence the system quickly becomes clogged with RETRIEVE commands waiting for dump tapes to be loaded before they can be implemented. A more efficient system for the retrieval of files has been developed by the Systems Software section based on software produced at RSRE Malvern and is in current use.

Errors

Message	Meaning
INCORRECT INCNO AND TSN COMBINATION	tsn incorrect for increment number specified
INCNO OK BUT TSN WRONG	tsn refers to an unsuitable magnetic tape, *eg* the tape may be unavailable
INCNO OK INC STATUS WRONG	specified increment exists but is not suitable

All three of the above errors will cause the macro to be terminated. 080

5.5 FSDICUPDATE

Function

The macro updates users' MONEY budgets according to the filestore charges computed by the SCANDUMP macro.

Format

FSDICUPDATE *CR file description, *LP filename

Both parameters are optional and may be omitted.

Description of the macro parameters

file description	the input file containing the filestore charge records. By default, the file used is :DUMPER.FILELIST
filename	the output file which is to receive records indicating success or otherwise of each user's budget update. By default a workfile is created to receive the output

Execution

This macro may only be run under :MANAGER because of the special budget extracodes used, *ie* peri type 60 and modes #40, #41, #50, #51 and #62.

As each account record is read in, the user's MONEY budget is updated and a record produced indicating the success or otherwise of the update. Records of all successful updatings are appended to the file FILERECS for processing at the end of the accounting period.

The following is a list of the codes used which indicate the result of an update attempt:

Code	Meaning
OK	a successful attempt update
-2	the user is a pseudo user
-4	no such user exists
-6	budget record not written as DICTIONARY would overflow

Errors

Message	Meaning
NG	command error on attempt to open the file DICTIONARY
DO	DICTIONARY almost full!

Both the above errors will cause the macro to be terminated.

5.6 LPCHARMAC

Function

LPCHARMAC calculates lineprinter paper charges and then updates the MONEY budgets in DICTIONARY. The macro is run once a week.

Format

LPCHARMAC FIN

The parameter FIN is optional.

Description of the macro parameter

FIN — indicates that the lineprinter paper charging run is the final one for that accounting period

Execution

The macro run under :JOURNAL, first checks for the existence of an LPCHARPROG file. If one exists then that indicates that the previous run of LPCHARMAC failed to run to completion possibly due to a George break. If, however, the previous LPCHARMAC run was successful then the macro loads a virgin copy of the charging program which takes as input the current TEMPJAPLIST file produced by JAPEXPANDMAC runs. Each record in TEMPJAPLIST consists of job identification (username etc) and the number of pages of lineprinter paper a LISTFILE issued by that job produced. The username and number of pages are extracted from each record and built-up into a list of usernames, each one followed by the total number of pages of lineprinter paper that user produced. Once this list is complete, a charge is computed for each user and the updating of MONEY budgets in DICTIONARY commences. The updating is done by the special budget extracode, peri type 60 mode #77.

The program is so written that each time the peri is issued a maximum of 25 MONEY budgets are updated. The program issues successive peris of this type for batches of up to 25 users until all the chargeable users' budgets have been updated.

Before each update, the current state of the program is preserved in a new generation of a file called LPCHARPROG and a new generation of the file PAPERCOST is assigned. Details of usernames and charges are written to the PAPERCOST files by the program.

During each update an update record is written to DICTIONARY containing the following information:

Word	Contents
0	record header
1	2
2	*UPD
3	ATE
4	LP01 - name of program creating this update record
5	1 - mark number of the program
6	time in seconds since midnight
7	date as days since 31 Dec 1899
8	unused
9	unused
10	unused
11	unused
12	LPCH
13	ARPR
14	OG(
15	generation number of LPCHARPROG containing latest SAVEd program
16	)

When all the updating has been completed then a new generation of the file TEMPJAPLIST is created, the data in the PAPERCOST files is appended to the file FORTPPERCOST and then the PAPERCOST files are erased.

If the FIN parameter was given, then a new generation of the file FORTPPERCOST is created; the previous generation of FORTPPERCOST then contains details of lineprinter paper charges for one accounting period. Finally, the LPCHARPROG files are erased.

If at the start of the macro run it was found that the previous LPCHARMAC run had not run to completion then the following action is taken. The program in LPCHARPROG is loaded and activated. It first reads (by issuing the special extracode type 60 mode #76) the update record in DICTIONARY and, from the information in this record, informs the macro via a HALTED message of the file description of the last LPCHARPROG file into which the charging program had been preserved. The macro then loads that program, activates it, and the program and macro continue to run as already described.

Errors

Message	Meaning
PROG FAILED IN DICUPDATE	program failure during the updating of DICTIONARY (*eg* a username no longer exists). The program will continue to run
PROG FAILED ??	unexpected program error has occurred. The macro is terminated
PROG FAILED IN SORTING	program error while sorting paper charge data. The macro is terminated

5.7 MTLIMITS

Function

The principle function of this macro is to generate a macro of ALLOWANCE commands based on the output from the budgetary control scheme. This generated macro is then later run under :MANAGER at the start of the next accounting period to replenish users' MONEY allowances.

Format

MTLIMITS

Description of the program parameters

The file IMCSPAR9 - XSDA parameters:

```
KEY03,(1,02A,0002.0;2,02A,0003.0;3,01A,0004.0,012),
NUM01,
INP(A)MT,
WORO1(C)ED,
WORO2(c)ED,
OUT(B=B)MT,
****
```

Execution

The macro runs under :ACCOUNTING, runs three programs which will now be briefly described.

The first takes as input computed MONEY allowances from the file CC ALLOWANCE generated by the budgetary control scheme software. The principal output is to a file called NEWALLOW and consists of a series of ALLOWANCE commands, one for each authorised user. Finally, the macro NEWALLOW is copied to

:MANAGER.IRENE.NEWALLOW. The secondary output, consisting of user name and allowance data, is to a magnetic tape file MTALLOW.

The second program reads the MTALLOW records and attaches standard sorting keys to each record. The extended records are written to a magnetic tape file KEYALLOW.

The third program is the standard ICL sorting program, XSDA, run under the macro XSDAMAC1. XSDA sorts the records into department, division and user name order, according to the parameters in the file IMCSPAR9, and outputs the sorted records to a magnetic tape file, SORTKEYALLOW. SORTKEYALLOW is later used as input to RUNACCOUNTSA.

Errors

Message	Meaning
M06A FAILED	unexpected program error occurred in the first program
M06B FAILED	unexpected program error occurred in the second program
XSDA FAILED	the type of error is defined by a message output by XSDA

All three of the above errors cause the macro to be terminated.

5.8 ENDFORTJOB

Function

This macro is run, under :JOURNAL, at the end of the accounting period to complete the processing of job records and do the final charging for lineprinter paper use.

Format

ENDFORTJOB

Execution

ENDFORTJOB first runs JAPEXPANDMAC. When JAPEXPANDMAC finishes, a new generation of the file JAPJOBS1906S (or JAPJOBS1904A) is created to hold the job records for the next accounting period.

Finally, LPCHARMAC FIN is run; this performs the final charging for lineprinter paper for the current accounting period.

Errors

080

As for JAPEXPANDMAC (section 5.3) and LPCHARMAC (section 5.6).

5.9 ACCOUNTJOB

Function

The macro has two main functions. The first is to collect accounting data from the DICTIONARY file and reset TIME allowances. The second is to give fresh MONEY allowances to each authorised user.

Format

ACCOUNTJOB

Execution

This macro, run under :MANAGER after ENDFORTJOB has been run, is the final accounting job to be run in an accounting period.

The macro first runs the standard ICL macro ACCOUNT[6] with parameter values as follows:

ACCOUNT IMULTO,IDIV1,LO,*LPIMCACCOUNT .

The ACCOUNT macro runs a program, XK7F, which reads serially through the records in the DICTIONARY file and takes the following actions for each set of budget records read:

(i) the allowance for each TIME budget is set equal to the ration,

(ii) all rations, new allowances (if any) and amounts consumed of TIME and MONEY budgets are sent to a file called ACCOUNTLP,

(iii) the amount consumed of each budget type is reset to zero, and

(iv) the total rations, allowances and amounts consumed of each TIME and MONEY budget for all users are sent to ACCOUNTLP.

No action is taken on SPACEMT and REALTIME budgets apart from their values being sent to ACCOUNTLP.

When the DICTIONARY has been processed the file ACCOUNTLP is copied to the file IMCACCOUNT and then listed and erased.

ACCOUNTJOB makes a further back-up copy of IMCACCOUNT in a new generation of the file BACKUP. ACCOUNTJOB then creates a new generation of the file FILERECS ready for the next accounting period's filestore charges.

Finally, the macro NEWALLOW is run. This macro is created by the macro MTLIMITS (see section 5.7) and contains a 'MONEY' ALLOWANCE command for each authorised user; thus users' MONEY budgets are replenished.

When ACCOUNTJOB has finished, George is ready to accept work for the new accounting period.

Errors

Refer to the ICL documentation[6] for the ACCOUNT macro.

5.10 M06XMAC

Function

This macro creates the direct access file DIVNAMES and stores in it all the division names with their corresponding department key numbers. The macro is run only when it is necessary to create or update DIVNAMES.

Format

M06XMAC

Description of the program parameters

The file IMCFORTPAR1.

Each line has the following format:

columns 1-4	division name or lodger unit code, *eg* MM1 or 612, left justified
columns 5-8	the corresponding department key number, right justified

The final line is followed by a line with asterisks in columns 1-4.

Execution

The macro, run under :ACCOUNTING, creates the direct access file DIVNAMES if it does not already exist, then loads a program which when activated reads the parameters from IMCFORTPAR1 and sends the division names and key numbers without further processing to DIVNAMES.

Errors

Message	Meaning
FAILED	an unexpected program error has occurred, the macro is terminated

5.11 M06YMAC

Function

This macro, run under :ACCOUNTING, creates the direct access file DEPTNAMES and stores in it all the department or lodger unit names with their corresponding
80 key numbers.

Format

M06YMAC

Description of the program parameters

The file IMCFORTPAR2.

Each line has the following format:

columns 1-12 department or lodger unit name, left justified, *eg* RADIONAV

columns 13-16 the key number for that department or lodger unit, right justified

The final line is followed by a line with asterisks in columns 1-4.

Execution

The macro creates the direct access file DEPTNAMES if it does not already exist, then loads a program which when activated reads the parameters from IMCFORTPAR2 and sends them without further processing to DEPTNAMES.

Errors

Message	Meaning
FAILED	an unexpected program error has occurred, the macro is terminated

5.12 RUNACCOUNTSA

Function

The macro processes the file :MANAGER.IRENE.IMCACCOUNT produced by the ACCOUNT macro run under ACCOUNTJOB (see section 5.9). The output, showing user spending in the last accounting period, is appended to similar output (on magnetic tape) for previous accounting periods in the quarter and will be later used in quarterly accounting. The output is also used by the budgetary control scheme.

Format

RUNACCOUNTSA a, b, AMEND

The parameters are in fixed positions. Parameter a may be null. Parameter AMEND is optional and may be omitted.

Description of the macro parameters

a — the tsn of the magnetic tape, ACTSSUMRY06S (or ACTSSUMRY04A) produced last time RUNACCOUNTSA was run. By default, no input magnetic tape required; this is the case at the start of each quarter

b — the tsn of the magnetic tape, ACTSSUMRY06S (or ACTSSUMRY04A) to receive output from this run

AMEND — indicates that one or more of the user accounts are to be amended, the amendments being input from the file IMCACCAMEND

Description of the program parameters

(a) The file IMCACCARD

Line 1

columns 1-16	dates of accounting period being processed, *eg* 20/06/7701/07/77
columns 17-18	generation number of the input magnetic tape, right justified, *eg* 06

Line 2

columns 1-8	date of Sunday preceding accounting period, *eg* 19/06/77

Line 3

columns 1-2	number of fields to be read from following line(s).

The fields in the following lines determine the order of output of department spending to the file ACCCPO.

Line 4, etc

columns 1-2	key number of first department
columns 3-4	key number of second department
columns 5-6	key number of third department

and so on

(b) The file IMCACCAMEND - amendments to user accounts

Each line

columns 1-12	user name, left justified, *eg* MM1164
column 15	+ sign or - sign; + means add to and - means subtract from the user's charge

column 16	£ sign - optional
columns 17-22	number of pounds to be added to or subtracted from the user's account

The last line is followed by a line with asterisks in columns 1-4. Note that the user names must be given in ascending department key number order and in ascending alphanumeric order within each department.

(c) The file IMCPAR4 - XSDA parameters

```
KEY04,(1,02A,0002.0;2,02A,003.0;3,01A,004.0,012),
KEY04,(4,02A,0007.0),
NUM01,
INP(A)MT,
WORO1(C)ED,
WORO2(C)ED,
OUT(B=B)MT,
****
```

Execution

The macro run under :ACCOUNTING, runs three programs, the functions of which are now briefly described.

The first takes as input the file :MANAGER.IRENE.IMCACCOUNT; each record read from this file is subsequently prefixed by sorting keys before being sent to a magnetic tape file, IMC4AACTS.

The records in IMC4AACTS are then sorted into department, division and username order, according to the parameters in the file IMCPAR4, by the second program, the standard ICL sorting program XSDA, run by the macro XSDAMAC1. The sorted records are sent to the magnetic tape file IMCSORTACTS.

The third program is then run. This first of all copies the data from the input magnetic tape (if any) to the output magnetic tape, then processes the file IMCSORTACTS as follows. The accounts records, in their department groupings, are sent to the lineprinter file IMCACCLIST, the money spent per user being modified when necessary by amendments read from the file IMCACCAMEND. The new money allowances, read from the magnetic tape file SORTKEYALLOW, are inserted in the account records prior to output. For each user, a record is written to the output magnetic tape, also called ACTSSUMRY06S (or ACTSSUMRY04A) but with a generation number one higher than the input magnetic tape, recording the money

spent in the last accounting period. Total mill used and money spent by each department is sent to a file IMCACCSUMA and to a file ACCCPO which is later used in performance analysis.

As an example, part of a listing of the file IMCACCLIST appears in Fig 22 and a listing of the file IMCACCSUMA appears in Fig 23.

Errors

Message	Meaning
MO6M FAILED	an unexpected program error has occurred in the first program
MO6N FAILED	an unexpected program error has occurred in the third program
XSDA FAILED	the type of error is defined by a message output by XSDA
B NOT SET	tsn for the output magnetic tape not given

All of the above errors will cause the macro to be terminated.

5.13 FORTPAPERMAC

Function

This macro takes as input the lineprinter paper charge records in :JAPFILES.FORTPPERCOST and sorts them into department and username order ready for use by NEWJAPJOBS (see section 5.15).

Format

FORTPAPERMAC

Description of the program parameters

The file FMO6ASORTPAR - XSDA parameters.

```
KEYO1(1,1A,2.0,12),
NUMO1,
INP(A)MT,
WORO1(C)ED,
WORO2(C)ED,
OUT(B=B)MT,
****
```

Execution

The macro, run under :ACCOUNTING, runs three programs the functions of which are now briefly described.

080

The first program takes as input each charge record in FORTPPERCOST and sends it to a magnetic tape file, PAPERRECS, prior to sorting.

The second program is the standard ICL sort program XSDA, run under the ICL macro XSDAMAC1. This sorts the records according to username, using the parameters in the file FM06ASORTPAR, and sends the sorted records to the magnetic tape file SORTPAPRRECS.

The third program reads the charge records from SORTPAPRRECS and sends one charge record (prefixed by the appropriate sorting keys) per user to the magnetic tape file PAPPROCRECS.

This final magnetic tape file will later be used by NEWJAPJOBS when the charge records will be sorted and merged with other types of charge records.

Errors

Message	Meaning
M06A FAILED 1 M06A FAILED 2	unexpected program error has occurred in first program - macro terminated
M06B FAILED	unexpected program error has occurred in the third program - macro terminated
XSDA FAILED	the type of error is defined by a message output by XSDA

All of the above errors will cause the macro to be terminated.

5.14 FILEACC

Function

This macro produces a listing showing spending on filestore by each user and each department in an accounting period. A magnetic tape file is also produced containing records of filestore spending to be used by NEWJAPJOBS (see section 5.15).

Format

FILEACC

Description of the program parameters

(a) The file IMCACCARD

Line 1

columns 1-16 dates of accounting period being processed, *eg* 20/06/7701/07/77

(b) The file IMCSPAR1 - XSDA parameters

```
KEY01(1,02A,0007.0),
NUM01,
INP(A)MT,
WOR01(C)ED,
WOR02(C)ED,
OUT(B=B)MT,
****
```

(c) The file IMCSPAR3 - XSDA parameters

```
KEY07,(1,01A,0002.0,012;2,01A,0007.2,002;3,01A,0006.3,002),
KEY07,(4,01A,0006.0,002;5,01A,12.0,2;6,01A,12.3,2),
KEY07,(7,01A,13.2,2),
NUM01,
INP(A)MT,
WOR01(C)ED,
WOR02(C)ED,
OUT(B=B)MT,
****
```

Execution

The macro run under :ACCOUNTING, runs six programs which are now briefly described.

The first program reads the filestore charge records from the file called :MANAGER.IRENE.FILERECS filled during the last accounting period and writes them to a magnetic tape file FSCHRECS.

The second program is the standard ICL sorting program XSDA, run under the macro XSDAMAC1. This sorts the records from FSCHRECS according to username, using the parameters in the file IMCSPAR3, and sends the sorted records to FSCHSORTRECS.

The third program reads the records from FSCHSORTRECS, totals the filestore spend for each user and sends one record per user to the magnetic tape file FSCHNEWREC. Before these records are sent the department number is attached to each record as the sorting key.

The fourth program XSDA sorts the records in FSCHNEWREC into department and user name order, using the parameters in the file IMCSPAR1, and sends the sorted records to FSCHNEWSORT.

The fifth program reads the records from FSCHNEWSORT and sends them, with appropriate headings and also with total spending on filestore output for each department, to the file IMCFSLP.

The sixth program reads the records from FSCHSORTRECS, and for each user writes to the magnetic tape file, FSCHPROCRECS, one record summarising spending by that user and his pseudo user inferiors for each working day throughout the accounting period. Each record produced has attached to it standard sorting keys. The file FSCHPROCRECS is later used by the macro NEWJAPJOBS (see section 5.15).

Part of a listing of the file IMCFSLP appears in Fig 24 as an example.

Errors

Message	Meaning
M06R FAILED 1 M06R FAILED 2	unexpected program error occurred in the first program
M06S FAILED	unexpected program error occurred in the third program
M06T FAILED	unexpected program error occurred in the fifth program
M06S(/MK01)FAILED	unexpected program error occurred in the sixth program
XSDA FAILED	the type of error is defined by a message output by XSDA

The macro is terminated if any of the above errors is encountered.

5.15 NEWJAPJOBS

Function

This macro sorts and merges together the job charge records in JAPJOBS1906S (or JAPJOBS1904A), the filestore charge records in FSCHPROCRECS and the paper charge records in PAPPROCRECS. A listing is finally produced showing the pattern of spending by each user throughout an accounting period.

Format

```
NEWJAPJOBS a,b  .
```

The parameter a is mandatory.

Description of the macro parameter

a	generation number of the JAPJOBS1906S (or JAPJOBS1904A) to be processed first
b	generation number of the JAPJOBS1906S (or JAPJOBS1904A) to be processed last

Description of the program parameters

(a) The file IMCACCARD

Line 1

columns 1-16 dates of accounting period being processed, *eg* 20/06/7701/07/77

(b) The file IMCSPAR2JAP - XSDC parameters

```
XSDC,KEY07(1,01A,0004.0,012;2,01A,11.2,002;3,01A,0010.3,002),
XSDC,KEY07(4,01A,0010.0,002;5,01A,0014.0,002;6,01A,0014.3,002),
XSDC,KEY07(7,01A,0015.2,002),
XSDC,NUM01,
XSDC,INP(FORTJOBRECS)MT,
XSDC,WORO1(SCRATCH FILE)(8000),
XSDC,WORO2(SCRATCH FILE)(8000),
XSDC,OUT(SORTFORTRECS)MT,
XSDC,
****
```

(c) The file FICHESORTPAR - XSDC parameters

```
XSDC,KEY10(0,2A,2.0;1,2A,3.0;2,1A,7.0,12),
XSDC,KEY10(3,2A,4.0;4,1A,14.2,2;5,1A,13.3,2),
XSDC,KEY10(6,1A,13.0,2;7,1A,17.0,2;8,1A,17.3,2),
XSDC,KEY10(9,1A,18.2,2),
XSDC,NUM03,
XSDC,INP(FSCHPROCRECS)MT,
XSDC,INP(JOBPROCRECS)MT,
XSDC,INP(PAPPROCRECS)MT,
XSDC,WORO1(SCRATCH FILE)(8000),
XSDC,WORO2(SCRATCH FILE)(8000),
XSDC,OUT(MAINFICHRECS=MAINFICHRECS)MT,
XSDC,
****
```

Execution

The macro, run under :ACCOUNTING, runs five programs which are now briefly described.

080 The first program reads the records from the series of JAPJOBS1906S (or JAPJOBS1904A) files for the accounting period, starting with the file with the

generation number a . The final file for the accounting period is terminated by a record with asterisks in columns 1 to 4. The job records are written to a magnetic tape FORTJOBRECS.

The second program is XSDC, a standard ICL sorting program. This program reads and sorts the job records from FORTJOBRECS into username, start date and start time order, using the parameters in the file IMCSPAR2JAP. These sorted records are then sent to the magnetic tape SORTFORTRECS.

The third program reads the sorted job records from SORTFORTRECS and attaches to each record standard sorting keys, *ie* keys to identify department, record type number, etc. These enlarged records are written to the magnetic tape JOBPROCRECS.

The fourth program, XSDC, sorts and merges the job charge records from JOBPROCRECS, the filestore charge records from FSCHPROCRECS and the paper charge records, using the parameters in the file FICHESORTPAR, from PAPPROCRECS. The sorted records are sent to the magnetic tape MAINFICHRECS. The records are sorted in order of department, division, username, record type, start date and start time.

The fifth program reads the records from MAINFICHRECS and writes them, with appropriate headings and sub-totals of CPU time and charges, to the file JOBLP. JOBLP is then listed and these listings subsequently distributed to users. The file JOBLP then goes through a process[5] which results in the accounts information being produced on microfiche.

Part of a listing of the file JOBLP appears in Fig 25 as an example.

Errors

Message	Meaning
M062 FAILED 1 M062 FAILED 2	execution error occurred in the first program
M53P FAILED	execution error in third program
XSDC ERROR IN LOADING	XSDC failed to load - corrupt program?
XSDC - UNEXPECTED EVENT	failure defined by an error code output by XSDC
M062 FAILED	execution error in the fifth program

All of the above errors will cause the macro to be terminated.

5.16 QUARTACC

Function

This macro takes as input the final ACTSSUMRY06S (or ACTSSUMRY04A) magnetic tape produced in a quarter and produces a listing showing for each user the total money spent throughout a quarter.

Format

QUARTACC a

Parameter a is mandatory.

Description of the macro parameter

a — the tsn of the final ACTSSUMRY06S (or ACTSSUMRY04A) produced in a quarter

Description of the program parameters

(a) The file QDATE

Line 1

columns 1-16 — dates of the quarter being processed in the form dd/mm/yy, *eg* 28/03/7701/07/77

Lines 2-8

columns 1-16 — dates of each accounting period in the quarter, in the form dd/mm/yy

NB: If there are only six accounting periods in a quarter, then line 7 is left blank.

(b) The file IMCSPAR8 - XSDA parameters

```
KEY03(1,02A,0011.0;2,02A,0012.0;3,01A,0006.0,012),
NUM01,
INP(A)MT,
WOR01(C)ED,
WOR02(C)ED,
OUT(B=B)MT,
****
```

Execution

The macro, run under :ACCOUNTING, runs three programs which will be briefly described.

The first program reads the account records from ACTSSUMRY06S (or ACTSSUMRY04A) and attaches sorting keys to each record. These enlarged records are then sent to a magnetic tape file NEW SUMMARY prior to sorting.

The second program run is the standard ICL sorting program XSDA run under macro XSDAMAC1. This program sorts the records into department groupings, according to the parameters in the file IMCSPAR8, the sorted records being then sent to the magnetic tape file IMCSORT1.

The third program then reads the sorted records and produces a listing for each department showing:

(a) total money spent by each user per accounting period and for the quarter, and

(b) grand totals of money spent by all users in that department per accounting period and for the quarter.

Part of this listing is shown in Fig 26 as an example.

Errors

Message	Meaning
M068 FAILED	unexpected program error occurred in the first program
M069 FAILED	unexpected program error occurred in the third program
XSDA FAILED	the type of error is defined by a message output by XSDA

The macro is terminated if any of the above three errors is encountered.

5.17 QUARTUPDATE

Function

This macro appends the latest quarterly accounts to a magnetic tape file containing the past quarters' account records for the current calendar year.

Format

QUARTUPDATE a

Parameter a is mandatory.

Description of the macro parameter

a the generation number of the magnetic tape file SUMMARY1906S (or SUMMARY1904A) containing past quarters' account records

Description of the program parameter

The file QUARTGEN

Line 1

columns 1-4 generation number of the input SUMMARY1906S (or SUMMARY1904A) file, right justified

Execution

The macro, run under :ACCOUNTING, runs a program which copies the contents of SUMMARY1906S (or SUMMARY1904A), generation a , to a newly created magnetic tape file SUMMARY1906S (or SUMMARY1904A), generation a + 1 , and then appends the latest quarter's account records to it.

If a has the value 0, then this indicates that there are no previous quarterly accounts to be copied, *ie* if January to March quarterly accounting is being done.

Errors

Message	Meaning
M066 FAILED	program error
%A NOT SET	parameter omitted

The macro is terminated if either of the above errors is encountered.

5.18 QUARTCOPY

Function

This macro makes a copy on a magnetic tape of the file SUMMARY1906S (or SUMMARY1904A) produced by QUARTUPDATE (see section 5.17). This is purely a safety measure to protect the quarterly accounts in the event of the latest SUMMARY1906S (or SUMMARY1904A) file being lost or corrupt.

Format

QUARTCOPY a,b

Both parameters are mandatory.

Description of the macro parameters

a the tsn of the magnetic tape to receive the output

b the generation number of the SUMMARY1906S (or SUMMARY1904A) file to be copied

Execution

This macro, run under :ACCOUNTING, runs a program which copies the contents of the specified SUMMARY1906S (or SUMMARY1904A) file to the specified magnetic tape.

Errors

Message	Meaning
M065 FAILED	program error
%A NOT SET	tsn parameter null
%B NOT SET	generation number of SUMMARY1906S (or SUMMARY1904A) omitted

6 CONCLUSION

This Report describes the mechanisms and software for accounting computer resources under the George 3 and 4 operating systems. It is expected that the general reader will find it sufficient to read through the first three sections only in order to acquire a general understanding of the accounting system. Sections 4 and 5 will mainly be of interest to those directly involved in the maintenance and operation of the system.

Although the accounting system is relatively complex and processes large amounts of data per day, it imposes a minimal overhead on the normal workload on each computer. It can be regarded, therefore, as a most economical method of monitoring the use of the computers. The accounts provide a firm basis for resource management and together with other data produced by the accounts suite they provide valuable data for performance analysis.

Appendix A

CHARGING ALGORITHMS

The charging algorithms are summarised below:

Job charging

The charge per job is given by:

$$\sum_{u=A}^{Z} \frac{M_u C_u}{i} + Ee + Tt$$

where M_u = total number of seconds of CPU time clocked at urgency u

C_u = charge per second of CPU time at that urgency

i = 1 if the job started in the period 08.00.00 to 17.59.59

i = 2 if the job started in the period 18.00.00 to 17.59.59 (next day)

E = the total connect time in minutes

e = the charge per minute of connect time

T = the total number of magnetic tape on-lines

t = the charge per magnetic tape on-line

Current charges

For CPU time

	1904A	1906S
C_J =	1.7p	8p
C_M =	1.4p	6p
C_Q and C_Z =	1.0p	3p

The value of C_u for other urgency levels is worked out according to the following algorithm:

$$\text{factor} \times (27 - \alpha)$$

	1904A	1906S
where factor =	0.001	0.0043

α = 1 for urgency A
2 for urgency B, etc .

For connect time

080 e = 5p for 1904A and 1906S .

For magnetic tape on-lines

$$t = 40p \qquad \text{for 1904A and 1906S} \ .$$

NB: There is a minimum charge of £1 per job.

Daily filestore charging

The charge per directory is given by

$$Nn + Ss$$

where N = total number of terminal files in the directory
n = charge per file
S = total size of all terminal files in the directory
s = charge per block.

Current charges

$$n = 3p \qquad \text{for 1904A and 1906S}$$

$$s = 0.25p \qquad \text{for 1904A and 1906S} \ .$$

NB: There is a minimum charge of £1 per directory for each directory containing at least one terminal file.

Lineprinter paper charging

The charge per user is given by

$$L\ell$$

where L = total number of pages output by LISTFILEs either at the central site or at the Maths 7020
ℓ = charge per page.

Current charges

$$\ell = 7p \qquad \text{for 1904A and 1906S} \ .$$

NB: There is no minimum charge.

Appendix B

FILESTORE

Every file in filestore is owned by a 'user' who has a unique username. The user may be a 'proper' user, *ie* a user with its own budgets, or a 'pseudo' user, *ie* a user who has no budgets or privileges of its own but shares those of a proper user.

The details of a user's files are recorded in the directory file associated with the username. A user may own two types of files, terminal files (containing programs, data or George commands) and directory files, each directory file being associated with a proper or a pseudo user. These inferior users may in turn own terminal files and further directory files, thus an hierarchical structure is built up.

Fig 3 shows part of the filestore structure as implemented at RAE. At the top of the hierarchy is the user :MASTER who is superior to the users :SYSTEM and :MANAGER. :SYSTEM and all its inferiors are concerned solely with the operational aspects of George, *eg* dumping files.

The structure under :MANAGER reflects the organisation of RAE departments. The usernames in this part of filestore are based on department or division names or on RAE costcodes. Each costcode relates to a particular project. The only jobs which may be run at department or division level are administrative jobs, *eg* jobs transferring budgets. Development and production jobs are run at costcode level or below.

The hierarchical structure can also be regarded as a 'management' structure. All users are ultimately under the control of :MANAGER and at lower levels under the control of their immediate superiors, *eg* a user at division level is able to change the budgets of an inferior user at costcode level.

Appendix C

BUDGETS

Budget records

Every proper user in filestore has an entry in a system file called DICTIONARY in which is recorded details of his budgets.

There are a number of budget types, but for this Report it is only necessary to describe the MONEY budget. The MONEY budget record for any user contains the following information.

(i) the current ration - at RAE always set to zero for costcode users and their inferiors;

(ii) the current allowance - money which can be spent immediately; and

(iii) the money spent so far by this user in this accounting period.

The allowance for each costcode user is calculated on the basis of money spent and the money left in his budget for the current financial year. The computed allowances are then distributed to the appropriate users at the start of the accounting period.

Periodically, accounting programs are run and charges for computer use calculated. These charges are added to the 'money spent' part of the appropriate user's MONEY budget records.

Whenever a job is introduced into the system, George performs checks for username and sometimes password validity and also checks the MONEY allowance against the total MONEY spent. If a user has spent more than his allowance then George rejects his jobs until he is back in credit.

At the end of each accounting period, a program is run which processes the budget records, printing out, amongst other things, the money spent by each user.

If a user is deleted from filestore, then all his budgets which of course include his debts, are inherited by his superior. The superior is thus accountable for all money spent by himself and any inferiors deleted before the end of the accounting period.

Transferring money

Money may be transferred from one user to another by means of the George command ALLOWANCE:

ALLOWANCE :username,MONEY,quantity .

The effect of this is either to increase the allowance of MONEY of any other user or decrease the allowance of MONEY of an immediately inferior user, *eg*

ALLOWANCE ;MM1164-A,MONEY,-2

if issued from :MM1164 has the effect of reducing the MONEY allowance of :MM1164-A by £2 and subsequently increasing the MONEY allowance of :MM1164 by £2.

Note that at RAE, one unit of MONEY represents £1.

The above command will only be accepted if there are sufficient funds in :MM1164-A's MONEY budget to cover the transfer of £2 without causing an overdraft.

Checking money budgets

During an accounting period a user may wish to know how much MONEY he or an inferior has left to spend; he can do this by means of the BUDGETQUERY George command:

BUDGETQUERY :username,MONEY .

The effect of this is that George will print out the MONEY status of the named user who must be an immediately inferior user. If no username parameter is given, then the MONEY status of the user himself is given, *eg*

BUDGETQUERY :MM1164-A,MONEY

may be issued by :MM1164 to find out how much MONEY :MM1164-A has left and how much has been spent.

REFERENCES

No.	Author	Title, etc
1	T.R.H. Sizer	The case for budgetary control on a central computing facility. RAE Technical Memorandum Math 7311 (1974)
2	D. Morgan T.R.H. Sizer	A budgetary control scheme for computing work at RAE Farnborough. RAE Technical Memorandum Math 7502 (1975)
3	D. Morgan	An expenditure control method for computer service work at RAE. RAE Technical Memorandum Math 7308 (1973)
4	Irene M. Cummings	The implementation of accounting for programs run on the 1907 computer at RAE Farnborough. RAE Technical Memorandum Math 7007 (1971)
5	Irene M. Cummings	A 'computer output to microfilm' facility for George users. RAE Technical Memorandum Math-Comp 7706 (1978)
6	ICL	George 3 and 4 Operation Management. Technical Publication 4334
7	C. Graff	Modifications to the ICL George 4 operating system to record the amount of lineprinter paper used. RAE Technical Memorandum Math 7602 (1976)
8	ICL	Direct access sorting. Technical Publication 4111

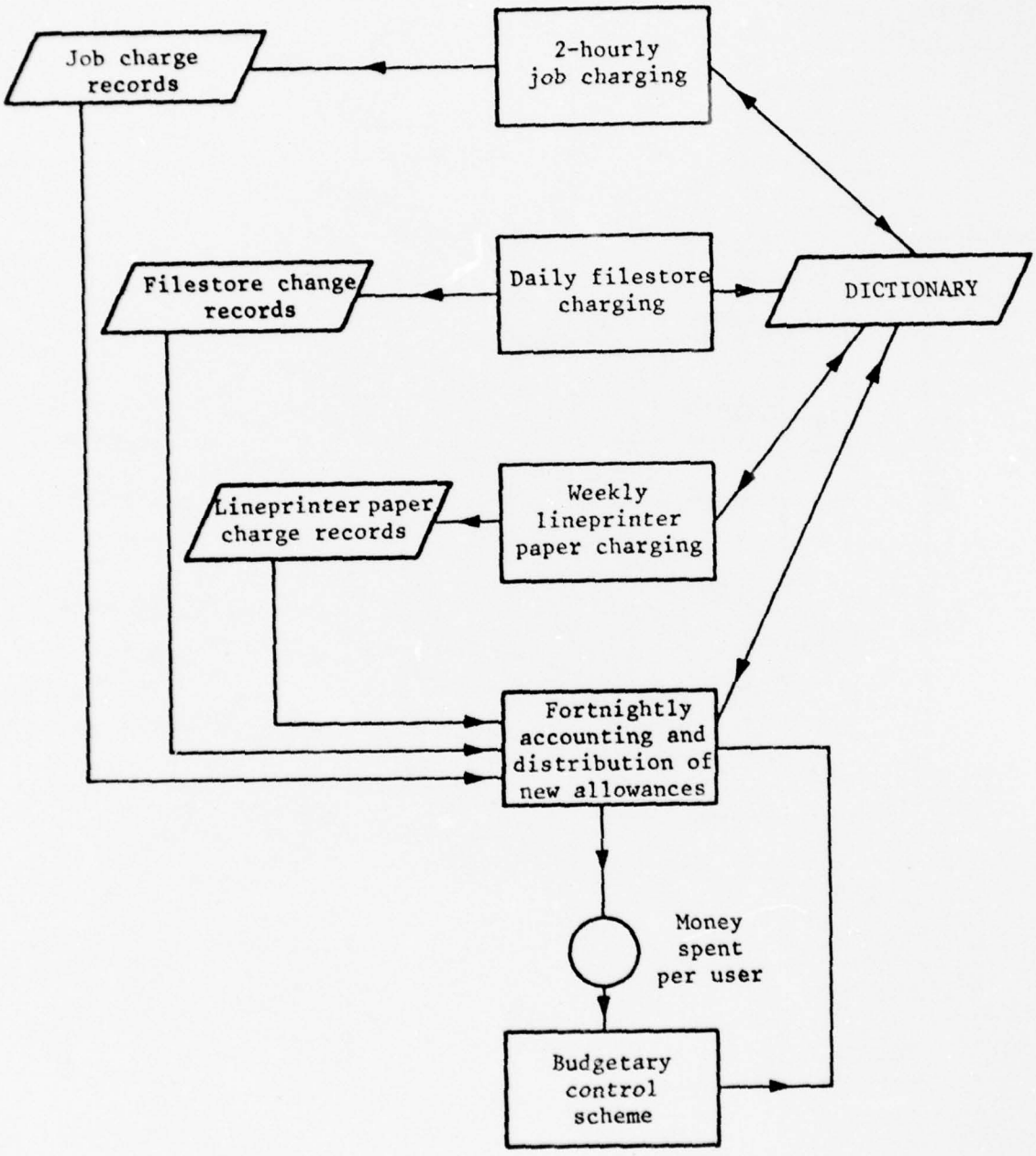


Fig 1 Interface with the budgetary control scheme

Fig 2

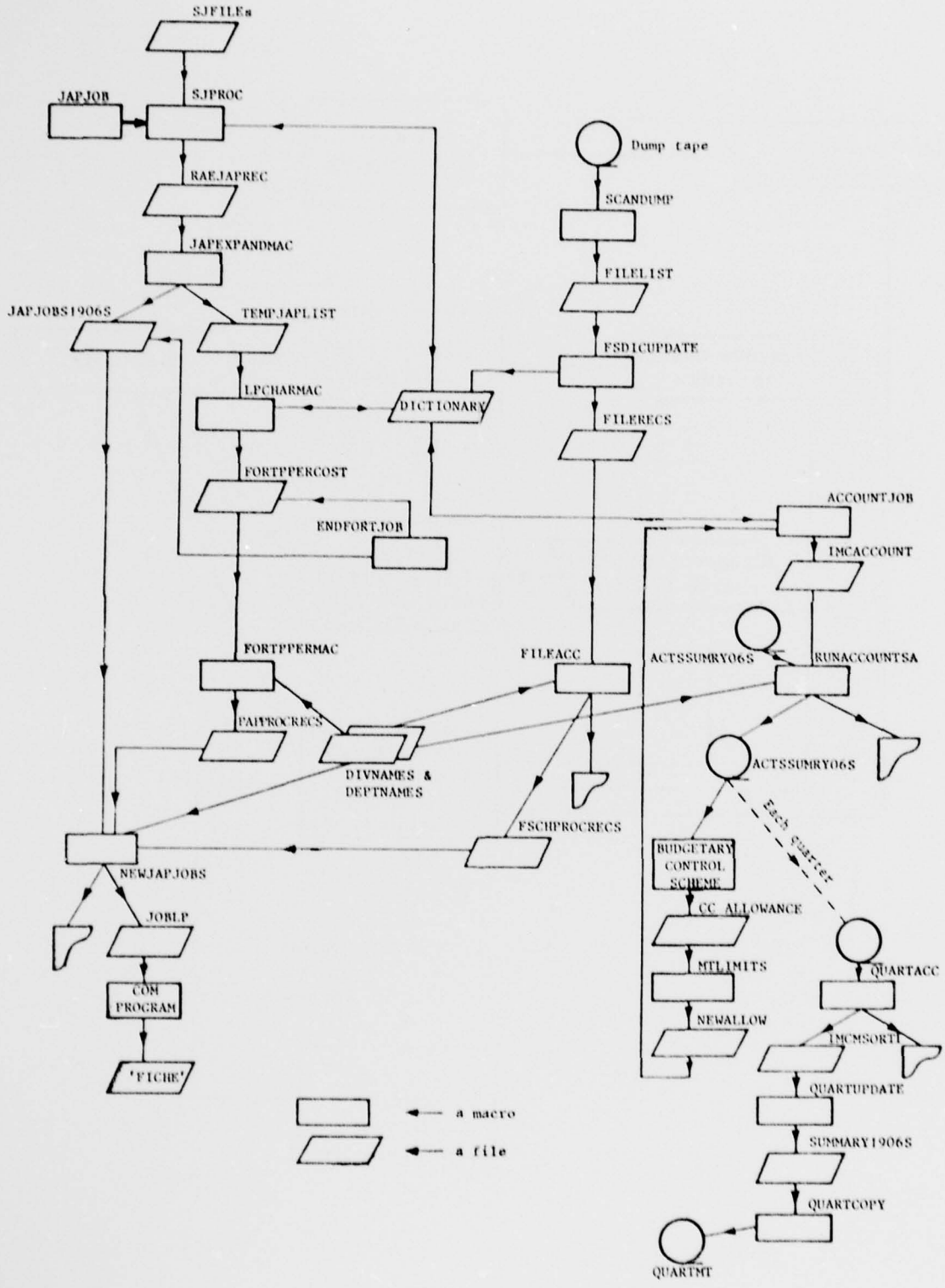


Fig 2 The George 3/4 accounting suite

TR 78080

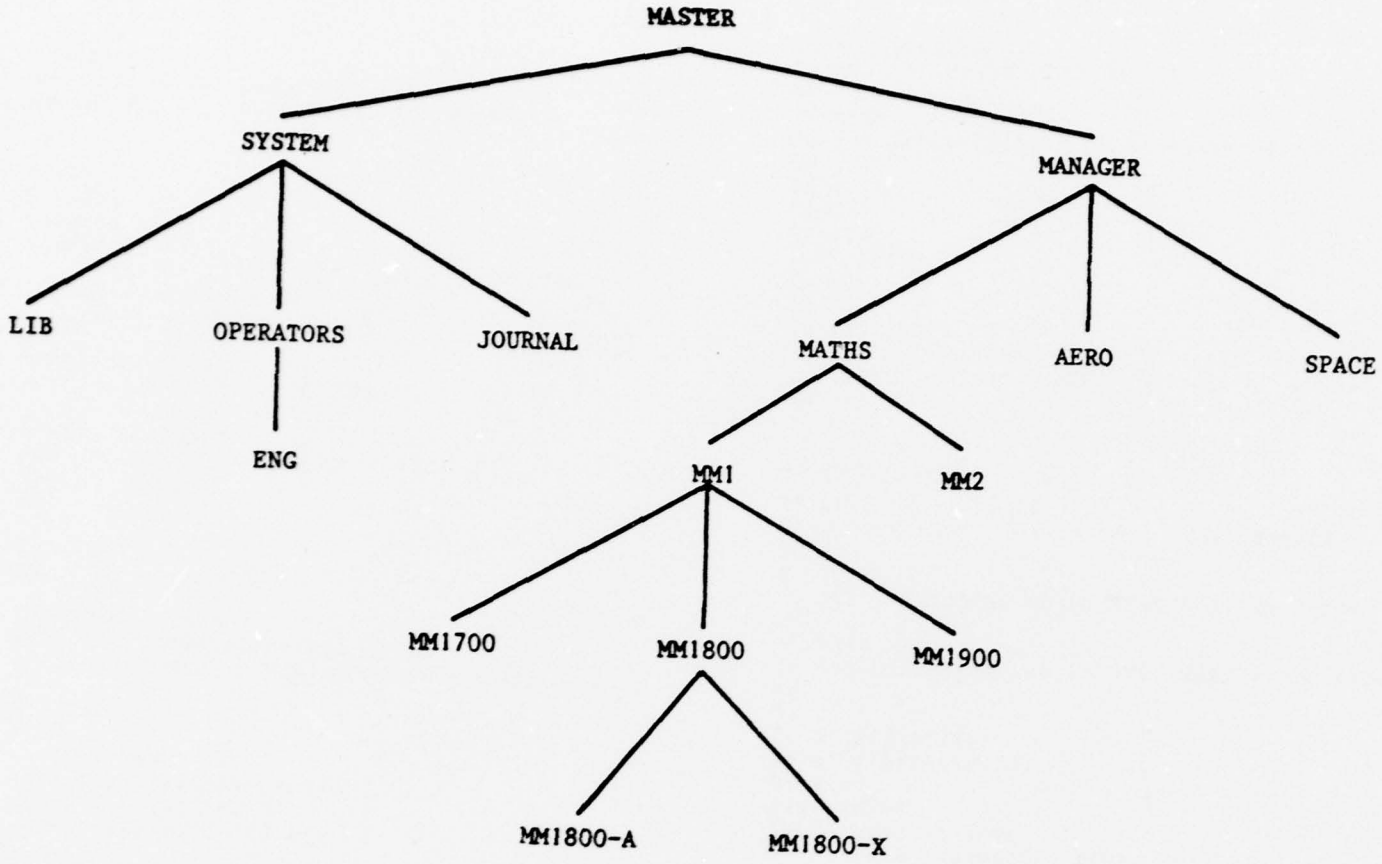


Fig 3 Part of the RAE filestore structure

```
9LOOP WT %A MINS
MS JAP STARTING
JAP %B,%C,%D,%E,%F,%G
GO 9LOOP
```

Fig 4 The JAPJOB macro

```
BY :JAPFILES
TRACE ALL
  IF STR(%Z)=(JOURNAL) AND STR(%Y)=(JAJOBA),GO 1GO
  DP 1,WRONG USER OR JOB NAME
  GO 99END
1GO  IF PRE(SJFILE),GO 1A
IF PRE(RAEJAPREC),GO 3A
#   NEITHER PARAMTER GIVEN
LO PROGRAM A68R
OL *LP0
TIME 10SECS
RM
IF HALTED(NO SUCH UPDATE RECORD),GO 40ERR
IF NOT HALTED(RAEJAPBIN),GO 41ERR
SP C,MESSAGE
DL
LO %C
TIME 5MINS
OL *LP0
ON 2
RM
IF NOT HALTED(RAEJAPREC),GO 42ERR
SP W,MESSAGE(11,14)
RM
IF NOT HALTED(SJFILE),GO 43ERR
SP X,MESSAGE(8,11)
GO 5A
#      SJFILE PARAMETER GIVEN
1A IF PRE(RAEJAPREC),GO 2A
#        SJFILE   GIVEN  BUT NOT   RAEJAPREC
SP X,(%(SJFILE))
LO PROGRAM A68R
TIME 5MINS
OL *LP0
ON 4
RM
IF HALTED(NO SUCH UPDATE RECORD),GO 10ERR
IF NOT HALTED(RAEJAPREC),GO 11ERR
SP W,MESSAGE(11,14)
RM
IF NOT HALTED(GEN NO OF SJFILE ?),GO 12ERR
AL 6,%X
OFF 4
GO 5A
#      BOTH PARAMETERS GIVEN
2A LO PROGRAM A68R
TIME 5MINS
OL *LP0
SP W,(%(RAEJAPREC))
SP X,(%(SJFILE))
ON 1
RM
IF NOT HALTED(GEN NO OF RAEJAPREC ?),GO 20ERR
AL 6,%W
RM
IF NOT HALTED(GEN NO OF SJFILE ?),GO 21ERR
AL 6,%X
GO 5A
#      RAEJAPREC  GIVEN  BUT  NOT  SJFILE
3A SP W,(%(RAEJAPREC))
LO PROGRAM A68R
TIME 10SECS
OL *LP0
RM
IF HALTED(NO SUCH UPDATE RECORD),GO 30ERR
IF NOT HALTED(RAEJAPBIN),GO 31ERR
SP C,MESSAGE
DL
LO %C
TIME 5MINS
OL *LP0
ON 2
ON 3
RM
IF NOT HALTED(GEN NO OF RAEJAPREC ?),GO 32ERR
AL 6,%W
RM
IF NOT HALTED(SJFILE),GO 33ERR
SP X,MESSAGE(8,11)
```

Fig 5 The SJPROC macro

Figs 4&5

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
OFF 3
5A AS *FR1,:JOURNAL.SJFILE(%X)(COMMUNE)
AS *CP0,:JAPFILES.RAEJAPREC(%U)(TS(ERASE))
9A  RN
IF HALTED(SAVE RAEJAPBIN),GO 6A
IF HALTED(RAEJAPREC),GO 7A
IF HALTED(SJFILE),GO 8A
IF HALTED(CAUGHT UP WITH GEORGE),GO 9A
IF DELETED(OK),GO 9END
GO 60ERR
6A SP 0,MESSAGE
%D
GO 9A
7A SP W,MESSAGE(11,14)
AS *CP0,:JAPFILES.RAEJAPREC(%U)(TS(ERASE))
GO 9A
8A SP X,MESSAGE(8,11)
AS *FR1,:JOURNAL.SJFILE(%X)(COMMUNE)
GO 9A
10ERR DL
DP 0,NO UPDATE RECORD
GO 99END
11ERR DL
DP 0,RAEJAPREC REQUIRED NOT SHOWN !
GO 99END
12ERR DL
DP 0,REQUEST FOR GEN NO OF SJFILE NOT MADE
GO 99END
20ERR DL
DP 0,REQUEST FOR GEN OF RAEJAPREC NOT MADE
GO 99END
21ERR DL
DP 0,REQUEST FOR GEN OF SJFILE NOT MADE
GO 99END
30ERR DL
DP 0,NO UPDATE RECORD
GO 99END
31ERR DL
DP 0,ERROR IN UPDATE RECORD?
GO 99END
32ERR DL
DP 0,REQUEST FOR GEN OF RAEJAPREC NOT MADE
GO 99END
33ERR DL
DP 0, THE SJFILE REQUIRED NOT KNOWN
GO 99END
40ERR DL
DP 0,NO UPDATE RECORD
GO 99END
41ERR DL
DP 0,ERROR IN UPDATE RECORD?
GO 99END
42ERR DL
DP 0,THE RAEJAPREC REQUIRED NOT KNOWN
GO 99END
43ERR DL
DP 0,THE SJFILE REQUIRED NOT KNOWN
GO 99END
60ERR DL
DP 0,FAILED?
GO 99END
9END DP 0,SUCCESSFUL RUN
99END  IF USER AND NOT MOP, EJ NONE,RETAIN(JAPJOBMON)
```

Fig5cont'd

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

Fig 5 contd

```
DY :JAPFILES
TRACE ALL
#
IF ABS(START),GO 1A
SP C,VALUE(%(START))
IF ABS(END),GO 2A
SP E,VALUE(%(END))
#
2A  LO JAPEXPAND
GO 4A
#
1A  IF ABS(END),GO 3A
SP E,VALUE(%(END))
#
3A  LO JAPEXPAND
ON 1
RM
IF NOT HALTED(RAEJAPREC START),GO 9ERR
SP C,MESSAGE(17,20)
#
4A  SP D,VALUE(%C+1)
IF ZERO(%C-4095),SP D,(1)
IF EXISTS(RAEJAPREC(%D)),GO 7GO
EXIT
#
7GO  IF EXISTS(RAEJAPREC(%C)),GO 1GO
DP 0,FGN %C OUT OF RANGE
GO 5GO
#
1GO  AS *CP0,JAPJOBS1906S(-0)(APPEND,LIMIT12000)
AS *CP1,TEMPJAPLIST(-0)(APPEND,LIMIT12000,COMMUNE)
#
1GOGO AS *FR1,RAEJAPREC(%C)
1RM  RM
IF NOT FAILED(OUTPUT *CP0  FILE FULL),GO 2RM
RL *CP0
LF JAPJOBS1906S(-0),*LP
AS *CP0, JAPJOBS1906S(+1)(TG(:MANAGER,GROUP,READ))
GO 1RM
2RM IF NOT HALTED(END OF FILE),GO 9ERR
TS RAEJAPREC(%C),READ
TG RAEJAPREC(%C),ERASE
IF ZERO(%C-4095),SP C,(0)
SP C,VALUE(%C+1)
IF PRE(END),GO 3GO
SP D,VALUE(%C+1)
IF ZERO(%C-4095),SP D,(1)
IF EXISTS(RAEJAPREC(%D)),GO 4GO
GO 5GO
#
4GO IF EXISTS(RAEJAPREC(%C)),GO 1GOGO
DP 0,ERROR IN FGN %C
GO 5GO
#
3GO IF POSITIVE(%C-%E),GO 5GO
GO 4GO
#
5GO ON 2
RM
IF NOT HALTED(GEN.NO.  OF NEXT FILE?),GO 9ERR

AL 6,%C
RM
IF NOT HALTED(SAVE PROG),GO 9ERR
SAVE JAPEXPAND
DL
GO 9END
#
9ERR DP 0,PROGRAM ERROR
#
9END  LF JAPJOBS1906S(-0),*LP
LF TEMPJAPLIST(-0),*LP
```

Fig 6 The JAPEXPANDMAC macro

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
SP K,"0"
SP L,"0"
SP M,"0"
IF PRE(JUG),AS *CP1,JUGGERNAUT(LIMIT 10000)
IF PRE(JUG),SP L,(%(JUG))
IF PRE(OLD),AS *LP1,OLDFILES(LIMIT20000,MULTIPLE)
IF PRE(OLD),SP M,(%(OLD))
 IF PRE(ACC),AS *CP0,FILELIST(+1)(TG(:MANAGER,READ))
 IF PRE(ACC),AS *TR0,ACCDATA
 IF PRE(ACC),AS *LP0,ACCOUNT
IF PRE(ACC),SP K,"1"
AS *FR1,:SYSTEM.INCINDEX
OL *MT0,(%(TAPE))
RM ,PARAM(%(INC),%(TAPE),%K,%L,%M)
DP 1,%;MESS;
IF NOT PRE(ACC),GO 1A
LF ACCOUNT,*LP
 ER ACCOUNT
1A IF NOT PRE(JUG),GO 1B
JUGGERNAUT
1B IF NOT PRE(OLD),GO 1C
#  LF OLDFILES,*LP
1C EJ ALL
```

Fig 7 The SCANDUMPJDF job description

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
LOAD
[ 01M06L0000200000000000000099+0000000000000000000000L5)S
[606000000000000000000000004W+Z
[#000000000000000000000000000000000000000000#0000000000000000000000000000004L+M
[#00000 000000000000074L03P5%742174227423742474257426742774287429000000002#]:
[#0000?1P000000000000000000000000000000@000000001 001*0(000000001 00100 003DY6
[#0000200000000019P00MOPEN    READ PATREAD RECWRITERECCLOSE     -1  -2  -3VJ%=
[#00002   -4  -5  -6  -7  -8USERUPDATE OK?    CHARGE    5T90****7@0@00000011
[#00002@0000000000000000000000000007@010000000*002JAS *C10,:MASTER.DICTIONAJ42+
[#00002PRY  7@0A00007@0H000000000000                              7@0I00009AY.
[#0000300000                                    7@0R000000000000000000000MONEY   D+;I
[#00003 0@1400002OK  @M1A    6X036X046X021L1)6$1[6@03<01<£01N=(02<018£01PRA %
[#00003@=(02<010E00N=(0"T00IP@0M6$1+6@023@5880Z%8@00<02&£00R=(03<02)£00VVHHA
[#00003P=(02T00BP@0M6$1+6@026$1[6@0380109(2-5P54&010,01£%(03D000L000805 %,HX
[#000040B=00; 40H@1)1L1'1L1(;@5>,01E,01),01'3@58001(2@4) 81(001'4<01205!QC)L
[#00004 ,123D00R-(02"P4%3P4)£<01(22=(@0T3P4+803"8@0RT00AP@0M,01£D00N-(03U]A'
[#00004@80148@0W6$1+6@02801(<402:@52802+8@1*<01*£01*=(0#<01££01*=(03<01N $EL
[#00004P£01.=(02<01P£01D=(025£01H01)T000J 2.T401803#$000R-00; 4$5£00R-00$VJE
[#0000506$1W6@041L1)3P3U6D036D026D04'40K803£8@0M<00M£00N=(0"3H008@1T1=001V$A
[#00005 ,300D02@-(031=0103000@101=02,300(@1R6$2F002G4@ 42X6"6$2.002/2H>$'Y>/
[#00005@DU3>L03;E(026$380039ZH5+003:0@1V0@2U6$2S002TE001!(00SP69ZH60X01UJ4<+
[#00005PL02UT03UM(08XD346$2+002+2H646$2Q002R2H62<0003P6%<0013P6%<0023P6%$(6G
[#000060<0033P6%<0053P6% 000E406"@6#<0043P6%1L30T030$051U(07H01VH@31H01U+£ H
[#00006 H@343P5T'4NG74D03P6#U@06(01R8C00PC016D0$801T;HU3000000000000000:Q"C
[303000#4$+J
IF PRE(*CR),GO 1B
AS *CR0,:DUMPER.FILELIST(-0)
GO 1A
1B AS *CR0,%(*CR)
1A IF PRE(*LP),GO 1C
CE !
AS *LP0,!
GO 1D
1C IF EXISTS(%(*LP)),ER %(*LP)
AS *LP0,%(*LP)
1D AS *CP0,FILERECS(-0)(APPEND,TG(:MM1162-M06,APPEND,READ))
EN 0
IF DELETED(OK),GO 1E
IF DELETED(NG),GO 1J
IF FAILED(ILLEGAL INSTRUCTION),GO 1G
IF FAILED(FILE *CR0 EXHAUSTED),GO 1H
1G PRINT (0,383)
1H DELETE
1E IF PRE(*LP),LF %(*LP),*LP
IF NOT PRE(*LP),LF !,*LP
1J
```

Fig 8 The FSDICUPDATE macro

Fig 8

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
DY :JAPFILES
IF EXISTS(LPCHARPROG(-0)),(LO LPCHARPROG(-0))ELSE(GO 2A)
ON 1
RM
IF NOT HALTED(LPCHARPROG),GO 2A
SP T,MESSAGE
IF NOT EXISTS(%T),GO 2A
LO %T
ON 2
GO 1A
2A LO NEWPROG
AS *FR1,:JOURNAL.JAPFILES.TEMPJAPLIST(COMMUNE)
RM
IF NOT HALTED(RECORDS SORTED AND STORED),GO SERR
SAVE LPCHARPROG(1)
1A RM
IF HALTED(SAVE LPCHARPROG),GO 4A
IF HALTED(DICUPDATE FAIL),GO 1ERR
IF HALTED(END OF PROGRAM),GO 9END
GO 2ERR
4A SP T,MESSAGE
%T
GO 1A

1ERR DP 0,PROG FAILED IN DICUPDATE
GO 1A

2ERR DP 0,PROG FAILED ??
EXIT

SERR DP 0,PROG FAILED IN SORTING
EXIT
  9END IF PRE(FIN),DP 1,OK START :MANAGER ACCOUNT JOB NOW
RL *LP0

10END AS *CP0,TEMPJAPLIST(+1)
TS TEMPJAPLIST(-1),WRITE,APPEND,READ
SP B,"1"
1B COPY PAPERCOST(%B),FORTPPERCOST(-0)(APPEND)
ER PAPERCOST(%B)
SP B,VALUE(%B+1)
IF EXISTS(PAPERCOST(%B)),GO 1B
LF FORTPPERCOST,*LP

IF PRE(FIN), AS *CP0,FORTPPERCOST(+1)(TS(ERASE,WRITE))
IF PRE(FIN),TG FORTPPERCOST(-1),:MANAGER,READ,GROUP
IF PRE(FIN),TG FORTPPERCOST(-1),:ACCOUNTING,READ,APPEND
1C IF EXISTS(LPCHARPROG),(ER LPCHARPROG)ELSE(GO 10END)
GO 1C
10END IF NOT MOP,EJ ALL
```

Fig 9 The LPCHARMAC macro

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
IF EXISTS(NEWALLOW),ER NEWALLOW
IF EXISTS(MTALLOW),ER MTALLOW
LO IMCM06ACRBIN
AS *CR0,CC ALLOWANCE
AS *MT0,MTALLOW(WRITE)
AS *LP0,NEWALLOW
ON 9
EN 0
IF DELETED,GO 1IMC
DP 0,M06A FAILED
DL
LF NEWALLOW,*LP
EXIT
1IMC LF NEWALLOW,*LP
TG NEWALLOW,:MANAGER,READ,EXECUTE
COPY NEWALLOW,:MANAGER.IRENE.NEWALLOW
IF EXISTS(KEYALLOW),ER KEYALLOW
IF EXISTS(SORTKEYALLOW),ER SORTKEYALLOW
LO IMCM06BMTBIN
AS *MT0,MTALLOW(READ)
AS *MT1,KEYALLOW(WRITE)
OL *LP0
AS *DA3,DIVNAMES(1)(OVERLAY)
AS *DA4,DEPTNAMES(1)(OVERLAY)
ON 9
EN 0
IF DELETED,GO 2IMC
DP 0,M06B FAILED
EXIT
2IMC XSDAMAC1 *CRIMCSPAR9,IN*MTKEYALLOW,OUT*MTSORTKEYALLOW,ER3IMC
GO 9IMC
3IMC DP 0,XSDA,FAILED
9IMC
```

Fig 10 The MTLIMITS macro

```
TRACE ALL
DP 1,PSE WAIT UNTIL U GET OK BEFORE
DP 1,RUNNING MANAGER JOB
JAPEXPANDMAC
LO JAPEXPAND
AS *CP0,:JAPFILES.JAPJOBS1906S(+1)
DL
TS :JAPFILES.JAPJOBS1906S(-0),WRITE,ERASE
TG :JAPFILES.JAPJOBS1906S(-0),:MM1,GROUP,READ
TG :JAPFILES.JAPJOBS1906S(-0),:SYSTEM,GROUP,READ
TG :JAPFILES.JAPJOBS1906S(-1),:MM1,READ,GROUP
TG :JAPFILES.JAPJOBS1906S(-1),:SYSTEM,READ,GROUP
TG :JAPFILES.JAPJOBS1906S(-1),:ACCOUNTING,READ,APPEND
DY :JOURNAL
LPCHARMAC FIN
EJ ALL
```

Fig 11 The ENDFORTJOB macro

Figs 10&11

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
DP 1,PSE KILL :JOURNAL.JAJOBA
DP 1,*****************************************
DP 1,MANAGER ACCOUNTS JOB NOW RUNNING
DP 1,*****************************************
DP 1,PSE RECORD EVENT IN YOUR DIARY
DP 1,*****************************************
DP 1,THIS JOB MUST NOT BE RUN AGAIN
DP 1,UNDER  ANY  CIRCUMSTANCES !!
DP 1,UNTIL END OF NEXT ACCOUNT PERIOD
DP 1,IF JOB FAILS DO NOT RERUN BUT
DP 1,REPORT TO SUPERVISOR AND
DP 1,RECORD EVENT IN DIARY
DP 1,*****************************************
DY IRENE
ER IMCACCOUNT
ACCOUNT IMULT0,IDIV1,L0,*LPIMCACCOUNT
COPY IMCACCOUNT,BACKUP(+1)
TG IMCACCOUNT,:ACCOUNTING,READ
LO PROGRAM XK7F
AS *CP0,FILERECS(+1)(APPEND)
DL
TG FILERECS(-1),:ACCOUNTING,READ,APPEND
NEWALLOW
EJ ALL
```

Fig 12 The ACCOUNTJOB macro

```
RV IMCMO6XBIN,IMCFORTPAR1
IF EXISTS DIVNAMES,RV DIVNAMES
IF EXISTS DIVNAMES,GO 1GO
GE DIVNAMES(1)(*DA,BUCK1,KWOR1)
1GO  LO IMCMO6XBIN
AS *CR0,IMCFORTPAR1
AS *DA2,DIVNAMES(1)(OVERLAY)
ON 9
EN 0
IF NOT DELETED,DP 0,FAILED
EXIT
```

Fig 13 The MO6XMAC macro

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
RV IMCM06YBIN,IMCSORTPAR2
IF EXISTS DEPTNAMES,RV DEPTNAMES
IF EXISTS DEPTNAMES,GO 1GO
CE DEPTNAMES(1)(*DA,KWOR1,BUCK1)
1GO LO IMCM06YBIN
AS *CR0,IMCSORTPAR2
AS *DA2,DEPTNAMES(1)(OVERLAY)
ON 0
EN 0
IF NOT DELETED,DP 0,FAILED
EXIT
```

Fig 14 The MO6YMAC macro

```
OR 0
RV :MANAGER.IRENE.IMCACCOUNT,IMCM06NBIN,DIVNAMES(1)
RV DEPTNAMES(1),IMCPAR4,IMCM06NBIN,IMCACCARD
IF EXISTS(IMC4AACTS),ER IMC4AACTS
LO IMCM06MBIN
TIME 3MINS
AS *CR0,:MANAGER.IRENE.IMCACCOUNT
AS *MT0,IMC4AACTS(WRITE)
AS *DA3,DIVNAMES(1)(OVERLAY)
AS *DA4,DEPTNAMES(1)(OVERLAY)
ON 9
EN 0
IF DELETED,GO 1IMC
DP 0,M06M FAILED
GO 9END
1IMC IF EXISTS(IMCSORTACTS),ER IMCSORTACTS
XSDAMAC1  *CRIMCPAR4,IN*MTIMC4AACTS,OUT*MTIMCSORTACTS,ER6IMC
IF EXISTS(IMCACCLIST),ER IMCACCLIST
IF EXISTS(IMCACCSUM),ER IMCACCSUM
IF EXISTS(ACCCP0),ER ACCCP0
IF EXISTS(IMCACCSUMA),ER IMCACCSUMA
LO IMCM06NBIN
TIME 3MINS
IF PRE(AMEND),ON 0
IF PRE(AMEND),AS *CR1,IMCACCAMEND
AS *CR0,IMCACCARD
AS *DA4,DEPTNAMES(1)(OVERLAY)
AS *MT0,IMCSORTACTS(READ)
AS *MT3,SORTKEYALLOW(READ)
AS *LP0,IMCACCLIST(LIMIT 7000)
AS *LP1,IMCACCSUM
AS *LP2,IMCACCSUMA
AS *CP0,ACCCP0
IF STRING(%A)=(),GO 3IMC
DP 1,PSE LOAD OWNED TAPE %A
OL *MT1(READ),(%A)
3IMC IF STRING(%B)=(),GO 4IMC
DP 1,PSE LOAD OWNED TAPE %B
OL *MT2(WRITE),(%B)
ON 9
EN 0
IF DELETED,GO 5IMC
DP 0,M06N FAILED
GO 9END
4IMC DP 0, B NOT SET
GO 9END
6IMC DP 0,XSDA FAILED
GO 9END
5IMC LF IMCACCLIST,*LP
LF IMCACCSUMA,*LP
LF ACCCP0,*CP
9END IF COR,DL
```

Fig 15 The RUNACCOUNTSA macro

Figs 14&15

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
TRACE ALL
RV FICHEM06ABN,FICHEM06BBN,:LIB.PROGRAM XSDA,:MACROS,XSDAMAC1
RV :JAPFILES,FORTPPERCOST(-1),DEPTNAMES,DIVNAMES
LO FICHEM06ABN
OL *LP0
AS *CP0,:JAPFILES.FORTPPERCOST(-1)(APPEND)
EN 0
IF HALTED (55) ,GO 1A
DP 0,M06A FAILED 1
GO 9END

1A  RL *CP0
 AS *CR0,:JAPFILES.FORTPPERCOST(-1)
AS *MT0,PAPERRECS(WRITE)
RM
IF DELETED,GO 2A
DP 0,M06A FAILED 2
GO 9END

2A IF EXISTS(SORTPAPRRECS),ER SORTPAPRRECS
 XSDAMAC1 *CREM06ASORTPAR.IN=MTPAPERRECS,OUT=MTSORTPAPRRECS,ER 1ERR
IF EXISTS(PAPPROCRECS),ER PAPPROCRECS
LO FICHEM06BBN
ON 9
OL *LP0
AS *MT0,SORTPAPRRECS(READ)
AS *MT1,PAPPROCRECS(WRITE)
AS *DA4,DEPTNAMES(1)(OVERLAY)
AS *DA5,DIVNAMES(1)(OVERLAY)
EN 0
IF DELETED,GO 9END
DP 0,M06B FAILED
GO 9END

1ERR DP 0,XSDA FAILED

9END  IF NOT.MOP,EJ ALL
```

Fig 16 The FORTPAPERMAC macro

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
RV DIVNAMES(1),DEPTNAMES(1)
RV IMCM06TBIN,IMCSPAR1,:MACROS.XSDAMAC1,:LIB.PROGRAM XSDA
RV IMCSPAR3,IMCM06RBIN,IMCM06SBIN(/     ),IMCM06SBIN(/MK01)
UR 0
LO IMCM06RBIN
ON 9
TIME 5MINS
AS *CPU,:MANAGER.IRENE.FILERECS(-1)(APPEND)
EN 0
IF HALTED(77),GO 1A
DP 0,M06R FAILED 1
DL
GO 9END
1A RL *CP0
IF EXISTS(FSCHRECS),ER FSCHRECS
AS *CR0,:MANAGER.IRENE.FILERECS(-1)
AS *MT0,FSCHRECS(WRITE)
RM
IF DELETED,GO 1B
DP 0,M06R FAILED 2
DL
GO 9END
1B
IF EXISTS(FSCHSORTRECS),ER FSCHSORTRECS
XSDAMAC1   *CRIMCSPAR3,IN*MTFSCHRECS,OUT*MTFSCHSORTRECS,ER1C
LO IMCM06SBIN(/     )
ON 9
TIME 5MINS
IF EXISTS(FSCHNEWREC),ER FSCHNEWREC
AS *DA4,DEPTNAMES(1)(OVERLAY)
AS *DA5,DIVNAMES(1)(OVERLAY)
AS *MT0,FSCHSORTRECS(READ)
AS *MT1,FSCHNEWREC(WRITE)
EN 0
IF DELETED,GO 1D
DP 0,M06S FAILED
GO 9END
1C DP 0,XSDA FAILED
GO 9END
1D IF EXISTS(FSCHNEWSORT),ER FSCHNEWSORT
XSDAMAC1    *CRIMCSPAR1,IN*MTFSCHNEWREC,OUT*MTFSCHNEWSORT,ER 1C
IF EXISTS(IMCFSLP),ER IMCFSLP
LO IMCM06TBIN
TIME 5MINS
ON 9
AS *LP0,IMCFSLP
AS *MT0,FSCHNEWSORT
AS *CR0,IMCACCARD
EN 0
IF DELETED,GO 1E
DP 0,M06T FAILED
GO 9END
  1E LF IMCFSLP,*LP
9END IF COR.DL
 LO IMCM06SBIN(/MK01)
ON 9
TIME 5MINS
IF EXISTS(FSCHPROCRECS),ER FSCHPROCRECS
AS *DA4,DEPTNAMES(1)(OVERLAY)

AS *DA5,DIVNAMES(1)(OVERLAY)
AS *MT0,FSCHSORTRECS(READ)
AS *MT1,FSCHPROCRECS(WRITE)
EN 0
IF DELETED,EXIT
DP 0,M06S(/MK01) FAILED
```

Fig 17 The FILEACC macro

Fig 17

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
RV ICM062JAPBIN,:JAPFILES.JAPJOBS1906S(-1)
RV KATHSORTBIN,DEPTNAMES,DIVNAMES
RV :LIB.PROGRAM XSDC,FICHESORTPAR,IMCSPAR2JAP
RV IMCM06ZBIN(/MK01)
TRACE ALL
LO ICM062JAPBIN
TIME 5MINS
AS *TP0,:JAPFILES.JAPJOBS1906S(%B)(APPEND)
ON 9
EN 0
IF HALTED(77),GO 1A
DP 0,M062 FAILED 1
GO 9END

1A  RL *TP0
AS *TR0,:JAPFILES.JAPJOBS1906S(%A)
OL *MT0(WRITE),FORTJOBRECS
1AA RM
IF NOT FAILED(FILE *TR0 EXHAUSTED),GO 1AAA
SP A,(%A+1)
AS *TR0,:JAPFILES.JAPJOBS1906S(%A)
GO 1AA
1AAA IF DELETED,GO 1B
DP 0,M062 FAILED 2
GO 9END

1B LO :LIB.PROGRAM XSDC
RM
IF NOT HALTED(LD),GO 2ERR
AS *CR0,IMCSPAR2JAP
MONITOR ON,OPEN
EN 1
1X  IF HALTED(HH),GO 1C
IF NOT MONITOR,GO 3ERR
IF MONITOR(OPEN          *DA),GO 3X
IF MONITOR(OPEN          *MT8),GO 4X
IF MONITOR(OPEN          *MT3),GO 5X
3X MR
GO 1X

4X SP D,MESSAGE(50,61)
OL *MT8(READ),%D
MR
GO 1X

5X OL *MT3(WRITE),SORTFORTRECS
MR
GO 1X

1C  LO KATHSORTBIN
TIME 5MINS
OL *LP0
OL *MT0,SORTFORTRECS(READ)
 OL *MT2(WRITE),JOBPROCRECS
 AS *DA4,DEPTNAMES(1)(OVERLAY)
 AS *DA5,DIVNAMES(1)(OVERLAY)
 ON 9

 EN 0
 IF NOT DELETED,GO 1ERR
 LO :LIB.PROGRAM XSDC
 RM
 IF NOT HALTED(LD),GO 2ERR
 AS *CR0,FICHESORTPAR
 MONITOR ON,OPEN
 EN 1
 2A  IF HALTED(HH),GO 1NEXT
 IF NOT MONITOR,GO 3ERR
 IF MONITOR(OPEN          *DA),GO 3A
 IF MONITOR(OPEN          *MT8),GO 4A
 IF MONITOR(OPEN          *MT3),GO 5A
 GO 4ERR
 3A MR
 GO 2A
 4A SP D,MESSAGE(50,61)
 WE COMERR,GO 6A
 RV %D
 AS *MT8,%D(READ)
 MR
 GO 2A
 6A WE COMERR
  OL *MT8(READ),%D
 MR
 GO 2A

 5A  OL *MT3(WRITE),MAINFICHRECS
 MR
 GO 2A

 1ERR DP 0,M53P FAILED
 GO 9END
 2ERR DP 0,XSDC ERROR IN LOADING
 GO 9END
 3ERR DP 0,XSDC - UNEXPECTED EVENT
 GO 9END

1NEXT LO IMCM06ZBIN(/MK01)
TIME 5MINS
OL *MT0(READ),MAINFICHRECS
AS *LP0,JOBLP(MULTIPLE,LIMIT40000)
OL *LP1
AS *DA5,DEPTNAMES(1)(OVERLAY)
AS *CR0,IMCACCARD
ON 9
EN 0
IF NOT DELETED,GO 5ERR
LF JOBLP,*LP
GO 9END
5ERR DP 0,M06Z FAILED
9END  IF NOT MOP,EJ ALL,RETAIN(MYMONFILE)
```

Fig 18 The NEWJAPJOBS macro

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

Fig 18

```
RV DIVNAMES,DEPTNAMES
RV IMCM068BIN,IMCM069BIN,IMCSPARR.QDATE
RV :MACROS.XSDAMAC1,:LIB.PROGRAM XSDA
LO IMCM068BIN
TIME 5MINS
IF STRING(%A)=(),GO 1ER
IF EXISTS(NEW SUMMARY),ER NEW SUMMARY
DP 1,PSE LOAD OWNED TAPE %A
OI *MT0(READ),(%A)
AS *MT1,NEW SUMMARY(WRITE)
AS *DA3,DIVNAMES(1)(OVERLAY)
AS *DA4,DEPTNAMES(1)(OVERLAY)
ON 9
EN 0
IF DELETED,GO 1A
DP 0,M068 FAILED
GO 9END
1A IF EXISTS(IMCMSORT1),ER IMCMSORT1
XSDAMAC1   *CRIMCSPARR.IN=MTNEW SUMMARY,OUT=MTIMCMSORT1,ER2ER
IF EXISTS(QUARTLP).ER QUARTLP
LO IMCM069BIN
TIME 5MINS
ON 9
IF EXISTS(QUARTLP).ER QUARTLP
AS *CR0,QDATE
AS *LP0,QUARTLP
AS *MT0,IMCMSORT1(READ)
EN 0
IF DELETED.GO 2A
DP 0,M069 FAILED
GO 2A
1ER DP 0,NO TSN NO  SET
GO 9END
2ER DP 0,XSDA FAILED
GO 9END
2A IF QUARTLP,*LP
9END IF COR,DI
```

Fig 19 The QUARTACC macro

```
LO IMCM066BIN
TIME 5MINS
AS *MT0,IMCMSORT1(READ)
OI *LP0
AS *CR0,QUARTGEN
IF ZERO %A,GO 1IMC
IF STRING(%A)=(),GO 2IMC
AS *MT1.SUMMARY1906S(%A)(READ)
1IMC SP B,VALUE(%A+1)
AS *MT2,SUMMARY1906S(%B)(WRITE)
ON 9
EN 0
IF DELETED,GO 9END
DP 0,M066 FAILED
DI
GO 9END
2IMC DP 0,%A NOT SET
9END
```

Fig 20 The QUARTUPDATE macro

```
LO IMCM065BIN
TIME 5MINS
IF STRING(%A)=(),GO 1IMC
OI *MT1(WRITE),(%A)
OI *LP0
IF STRING(%B)=(),GO 2IMC
AS *MT0,SUMMARY1906S(%B)(READ)
ON 9
EN 0
IF DELETED.GO 9END
DP 0,M065 FAILED
DI
GO 9END
1IMC DP 0,%A NOT SET
DL
GO 9END
2IMC DP 0,%B NOT SET
DL
9END
```

Fig 21 The QUARTCOPY macro

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

ICL 1906S COMPUTER AT RAE FARNBOROUGH

GEORGE 4 FORTNIGHTLY ACCOUNTS FROM 07/11/77 TO 18/11/77

ACCOUNT FOR :KK1126 WHOSE SUPERIOR IS :KK1

BUDGET TYPE	NEW ALLOWANCE	CURRENT RATION	AMOUNT USED	UNIT TYPE		
$PACEMT		2	1	MT	CHARGE £	0
TIME(M)			22	MILLSEC		
TIME(Q)	300	300	70	MILLSEC		
MONEY	1972		182	£		

ACCOUNT FOR :KK1126-M23 WHOSE SUPERIOR IS :KK1126

BUDGET TYPE	NEW ALLOWANCE	CURRENT RATION	AMOUNT USED	UNIT TYPE		
$PACEMT		9	6	MT	CHARGE £	0
TIME(A)			39	MILLSEC		
TIME(M)			2422	MILLSEC		
TIME(Q)	2500	2500	2833	MILLSEC		
TIME(Z)			386	MILLSEC		
MONEY			932	£		
REALTIME		12000		WORDS		

ACCOUNT FOR :KK1126-M36 WHOSE SUPERIOR IS :KK1126

BUDGET TYPE	NEW ALLOWANCE	CURRENT RATION	AMOUNT USED	UNIT TYPE		
$PACEMT		1	1	MT	CHARGE £	0
TIME(M)			34	MILLSEC		
TIME(Q)	400	400	76	MILLSEC		
MONEY			87	£		

ACCOUNT FOR :KK1126-M39 WHOSE SUPERIOR IS :KK1126

BUDGET TYPE	NEW ALLOWANCE	CURRENT RATION	AMOUNT USED	UNIT TYPE
TIME(Q)	500	500		MILLSEC
MONEY			20	£

ACCOUNT FOR :KK1126-M45 WHOSE SUPERIOR IS :KK1126

BUDGET TYPE	NEW ALLOWANCE	CURRENT RATION	AMOUNT USED	UNIT TYPE		
$PACEMT		3	3	MT	CHARGE £	0
TIME(M)			39	MILLSEC		
TIME(Q)	500	500	148	MILLSEC		
MONEY			89	£		

Fig 22 Part of the file IMCACCLIST output by RUNACCOUNTSA

Fig 23

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

ICL 1906S COMPUTER AT RAE FARNBOROUGH

GEORGE 4 FORTNIGHTLY ACCOUNTS FROM 15/08/77 TO 26/08/77

DEPT	TOTAL TIME CLOCKED (SECS)	TOTAL MONEY SPENT (£)
AAAAAAAAAAAA	81054	5609
BBBBBBBBBBBB	10	26
CCCCCCCCCCCC	0	30
DDDDDDDDDDDD	577	405
EEEEEEEEEEEE	4394	3530
FFFFFFFFFFFF	3943	263
GGGGGGGGGGGG	6163	2504
HHHHHHHHHHHH	31477	3952
IIIIIIIIIIII	0	0
JJJJJJJJJJJJ	0	130
KKKKKKKKKKKK	6688	1576
LLLLLLLLLLLL	11236	1279
MMMMMMMMMMMM	1649	589
NNNNNNNNNNNN	4120	1453
OOOOOOOOOOOO	0	30
PPPPPPPPPPPP	0	0
QQQQQQQQQQQQ	2478	480
RRRRRRRRRRRR	106107	19459
SSSSSSSSSSSS	7379	1704
TTTTTTTTTTTT	2227	844
UUUUUUUUUUUU	10150	776
VVVVVVVVVVVV	3278	1178
WWWWWWWWWWWW	13797	2090
TOTALS	296733	47507

Fig 23 File IMCACCSUMA output by RUNACCOUNTSA

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

ICL 1906S COMPUTER AT RAE FARNBOROUGH

GEORGE 4 FILESTORE CHARGES FOR PERIOD 15/08/77 TO 26/08/77

USER NAME	TOTAL CHARGE(£)
XX1 [illegible]	23
XX1 [illegible]	80
XX2 [illegible]	10
XX2 [illegible]	20
XX2 [illegible]	28
XX2 [illegible]	10
XX3 [illegible]	10
XX3 [illegible]	50
XX3 [illegible]	10
XX3 [illegible]	10
XX3 [illegible]	50
XX3 [illegible]	10
XX3 [illegible]	20
XX3 [illegible]	10
XX DEPT	10
	351

Fig 24 File IMCFSLP output by FILEACC

DEPT AAAAAAAAAAAA ICL 1906S COMPUTER AT RAE FARNBOROUGH AR114X03 PAGE 1

GEORGE 4 ACCOUNTS FOR PERIOD 15/08/77 TO 26/08/77

USER NAME AR114X03

JOB CHARGES

JOB NAME	START DATE	FINISH DATE	START TIME	FINISH TIME	TIME(J)	TIME(M)	TIME(Q)	TIME(Z)	OTHER MILL	TOTAL MILL	CONNECT (MINS)	TOTAL CHARGE(£)	JOB TYPE
M-T87Z	15/08/77	15/08/77	09.31.00	10.10.57	0	1	20	0	0	21	40	3	MOP
M-T87Z	15/08/77	15/08/77	10.44.48	11.26.42	0	3	46	0	0	49	42	4	MOP
M-T53	15/08/77	15/08/77	13.57.04	14.22.42	0	1	43	0	0	44	13	2	MOP
M-T538	15/08/77	15/08/77	14.10.23	14.21.56	0	1	5	0	0	6	0	1	BACK
M-T53C	15/08/77	15/08/77	14.22.10	14.44.33	0	1	74	0	0	75	0	2	BACK
M-T87Z	15/08/77	15/08/77	15.52.19	16.34.37	0	1	18	0	0	19	42	3	MOP
M-T53	15/08/77	15/08/77	16.07.37	16.08.26	0	1	0	0	0	1	1	1	MOP
M-T53Z	15/08/77	15/08/77	19.51.14	20.17.21	0	1	39	0	0	40	0	1	BACK
M-T87Z	16/08/77	16/08/77	09.21.30	10.50.14	0	1	12	0	0	13	89	5	MOP
M-T53	16/08/77	16/08/77	11.34.42	13.00.44	0	1	227	0	0	228	0	10	BACK
M-T53	16/08/77	16/08/77	15.58.50	16.34.34	0	1	34	0	0	35	28	3	MOP
M-T87Z	16/08/77	16/08/77	16.01.49	16.37.31	0	1	2	0	0	3	36	2	MOP
M-T53	17/08/77	17/08/77	09.44.47	10.03.47	0	1	85	0	0	86	19	4	MOP
M-T87Z	17/08/77	17/08/77	10.47.26	12.01.20	0	4	90	0	0	94	74	7	MOP
M-T87Z	17/08/77	17/08/77	15.22.43	16.37.58	0	5	4	0	0	9	75	4	MOP
M-T87Z	18/08/77	18/08/77	10.39.40	11.40.01	0	1	106	0	0	107	60	6	MOP
M-T87Z	18/08/77	18/08/77	12.16.27	12.53.13	0	1	11	0	0	12	37	2	MOP
M-T87Z	18/08/77	18/08/77	14.25.31	16.14.54	0	8	0	0	0	8	109	6	MOP
M-T87Z	19/08/77	19/08/77	09.00.21	09.52.50	0	6	0	0	0	6	52	3	MOP
M-T53	19/08/77	19/08/77	09.03.33	09.05.55	0	1	0	0	0	1	2	1	MOP
M-T53	19/08/77	19/08/77	10.37.56	10.53.13	0	1	13	0	0	14	0	1	BACK
M-T53	19/08/77	19/08/77	14.22.15	14.34.13	0	1	11	0	0	12	12	1	MOP
M-T53	19/08/77	19/08/77	14.37.57	14.40.25	0	1	0	0	0	1	2	1	MOP
M-T87Z	22/08/77	22/08/77	11.26.18	11.42.51	0	1	9	0	0	10	17	1	MOP
M-T87Z	22/08/77	22/08/77	14.50.27	15.31.57	0	3	0	0	0	3	42	2	[illegible]
M-T53	23/08/77	23/08/77	08.44.28	09.51.27	0	1	54	0	0	55	67	6	[illegible]
M-T87Z	23/08/77	23/08/77	11.13.29	12.08.19	0	2	133	0	0	135	55	7	MOP
M-T87Z	24/08/77	24/08/77	09.34.32	09.56.36	0	2	8	0	0	10	22	1	MOP
M-T53	24/08/77	24/08/77	11.43.46	12.12.45	0	1	14	0	0	15	29	2	MOP
M-T87Z	24/08/77	24/08/77	14.33.06	14.57.54	0	4	0	0	0	4	25	1	MOP
M-T87Z	25/08/77	25/08/77	09.34.00	11.56.20	0	1	32	0	0	33	142	9	MOP
M-T87Z	25/08/77	25/08/77	15.33.41	16.37.37	0	1	35	0	0	36	64	4	MOP
M-T53	26/08/77	26/08/77	08.36.36	08.53.14	0	1	1	0	0	2	10	1	MOP
M-T53X	26/08/77	26/08/77	08.46.22	08.53.26	0	3	0	0	0	3	7	1	MOP
M-T87Z	26/08/77	26/08/77	10.34.22	11.25.45	0	8	9	0	0	17	51	3	MOP
M-T87Z	26/08/77	26/08/77	12.12.51	12.27.14	0	1	25	0	0	26	14	1	MOP
				TOTALS	0	73	1160	0	0	1233	1278	112	

Fig 25 Part of file JOBLP output by NEWJAPJOBS

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

DEPT AAAAAAAAAAAA ICL 1906S COMPUTER AT RAE FARNBOROUGH AR114X03 PAGE 2

GEORGE 4 ACCOUNTS FOR PERIOD 15/08/77 TO 26/08/77

USER NAME AR114X03

FILESTORE CHARGES

DATE	TIME	TOTAL CHARGE(£)
15/08/77	08.02.18	33
16/08/77	07.01.02	32
17/08/77	07.30.26	33
18/08/77	06.50.22	33
19/08/77	06.09.07	33
19/08/77	21.19.43	33
23/08/77	06.28.39	33
24/08/77	07.26.28	34
25/08/77	07.00.16	32
26/08/77	07.03.29	33
TOTAL		329

LINE PRINTER PAPER CHARGES

TOTAL NO OF PAGES	TOTAL CHARGE(£)
1862	131

Fig25cont'd

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

Fig 25 contd

ICL 1906S COMPUTER AT RAE FARNBOROUGH

TOTAL MONEY SPENT BY USERS IN PERIOD 03/07/78-06/10/78

MONEY SPENT PER ACCOUNTING PERIOD

USER NAME	03/07/78 14/07/78	17/07/78 28/07/78	31/07/78 11/08/78	14/08/78 25/08/78	28/08/78 08/09/78	11/09/78 22/09/78	25/09/78 06/10/78	TOTAL MONEY SPENT(£)
AL	0	0	0	0	0	0	0	0
AL1	0	0	0	0	0	0	0	0
AL2	0	0	4	0	0	1	1	6
AL3	0	0	0	0	1	0	0	1
	0	0	4	0	1	1	1	7

MONEY SPENT PER ACCOUNTING PERIOD

USER NAME	03/07/78 14/07/78	17/07/78 28/07/78	31/07/78 11/08/78	14/08/78 25/08/78	28/08/78 08/09/78	11/09/78 22/09/78	25/09/78 06/10/78	TOTAL MONEY SPENT(£)
AL101V071	39	30	41	116	18	18	20	282
AL101X04	40	40	36	40	36	36	40	268
AL1914	28	10	9	10	9	9	10	85
AL201U081	0	0	86	10	34	13	10	153
AL201U082	10	10	36	10	9	9	10	94
AL201V061	50	29	19	84	111	85	65	423
AL201V062	40	43	94	21	19	18	116	360
AL209M041	0	0	0	0	0	0	0	0
AL217S025	0	0	0	0	0	0	0	0
AL2201A	1	0	3	11	10	9	11	45
AL2913	0	0	0	0	0	35	49	84
AL301U061	0	0	0	0	0	0	0	0
AL301V031	0	0	0	0	90	29	99	218
AL301V032	40	40	36	51	27	27	30	251
AL301V033	0	0	0	0	0	0	0	0
AL301W011	20	20	18	20	18	18	20	134
AL301W02	48	89	82	60	54	54	53	440
AL301W031	0	0	0	0	0	0	0	0
AL301W032	14	10	24	10	9	9	10	86
AL301W051	0	0	0	0	0	0	0	0
	319	321	484	443	444	369	543	2923

Fig 26 Part of file QUARTLP output by QUARTACC

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

REPORT DOCUMENTATION PAGE

Overall security classification of this page

UNCLASSIFIED

As far as possible this page should contain only unclassified information. If it is necessary to enter classified information, the box above must be marked to indicate the classification, e.g. Restricted, Confidential or Secret.

1. DRIC Reference (to be added by DRIC)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification/Marking
	RAE TR 78080	N/A	UNCLASSIFIED

5. DRIC Code for Originator	6. Originator (Corporate Author) Name and Location
850100	Royal Aircraft Establishment, Farnborough, Hants, UK
5a. Sponsoring Agency's Code	**6a. Sponsoring Agency (Contract Authority) Name and Location**
N/A	N/A

7. Title The George 3 and 4 accounting system at RAE

7a. (For Translations) Title in Foreign Language

7b. (For Conference Papers) Title, Place and Date of Conference

8. Author 1. Surname, Initials	9a. Author 2	9b. Authors 3, 4	10. Date	Pages	Refs.
Cummings, Irene M.	-	-	July 1978	74	8

11. Contract Number	12. Period	13. Project	14. Other Reference Nos.
N/A	N/A		Math-Comp 233

15. Distribution statement
(a) Controlled by –
(b) Special limitations (if any) –

16. Descriptors (Keywords) (Descriptors marked * are selected from TEST)

Computing. Accounting.

17. Abstract

A comprehensive system for accounting for computer usage has been developed at RAE for running under the George Operating System on the ICL 1904A and 1906S computers. It allows any resource to be accounted for and has been running successfully for the last 3 years.

This Report describes the mechanism of the accounting system in detail and includes a discussion on the choice of resources actually accounted for and the charges made.

F5910/1

RAE Form A143